

PENGEMBANGAN APLIKASI PEMBELAJARAN PEMROGRAMAN JAVA YANG ATRAKTIF BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Riski Pradana
NIM: 145150200111199



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN**PENGEMBANGAN APLIKASI PEMBELAJARAN PEMROGRAMAN JAVA YANG
ATRAKTIF BERBASIS ANDROID****SKRIPSI**


Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer


Disusun Oleh :
Riski Pradana
NIM: 145150200111199

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II


Bayu Priyambadha, S.Kom, M.Kom
NIP: 19820909 200812 1 004


Fajar Pradana, S.ST, M.Eng
NIP: 19871121 201504 1 004

Mengetahui
Ketua Jurusan Teknik Informatika



Dr. Asfoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiaris, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018



Riski Pradana

NIM: 145150200111199

KATA PENGANTAR

Segala puji dan syukur bagi Allah Subhanahu Wa Ta'ala atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "PENGEMBANGAN APLIKASI PEMBELAJARAN PEMROGRAMAN JAVA YANG ATRAKTIF BERBASIS ANDROID".

Untuk kesempatan ini penulis juga menyampaikan rasa terima kasih kepada pihak-pihak yang telah membantu penulis selama penyusunan skripsi, diantaranya:

1. Allah Subhanahu Wa Ta'ala yang telah memberi kemudahan dalam semua proses penulisan skripsi ini.
2. Kedua orang tua penulis, yaitu Bapak Djiono dan Ibu Supin beserta keluarga besar yang selalu memberikan segala masukan, do'a, motivasi dan semangat yang tidak terputus.
3. Bapak Bayu Priyambadha, S.Kom, M.Kom selaku Dosen Pembimbing I yang telah meluangkan waktu untuk memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
4. Bapak Fajar Pradana, S.ST, M.Eng selaku Dosen Pembimbing II yang juga telah meluangkan waktu untuk memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
5. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.
7. Seluruh pihak yang terlibat dalam proses pengerjaan skripsi dalam bentuk apapun yang tidak bisa penulis tuliskan semuanya dalam kata pengantar yang singkat ini.

Penulis menyadari masih banyak kekurangan dalam penyusunan skripsi ini baik dalam teknik penyajian materi maupun pembahasan. Demi kesempurnaan penelitian skripsi ini, penulis sangat mengharapkan saran dan kritik yang sifatnya membangun. Besar harapan penulis melalui penulisan skripsi ini dapat memberikan sumbangsih terhadap perkembangan ilmu pengetahuan serta dapat memberikan manfaat bagi pihak yang membutuhkan.

Malang, 2 Agustus 2018

Penulis

riskipradana@student.ub.ac.id

ABSTRAK

Riski Pradana, Pengembangan Aplikasi Pembelajaran Pemrograman Java yang Atraktif Berbasis Android

Pembimbing: Bayu Priyambadha, S.Kom, M.Kom dan Fajar Pradana, S.ST, M.Eng

Di Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya, pemrograman dasar menjadi salah satu mata kuliah yang wajib ditempuh oleh seluruh mahasiswa. Pemrograman dasar di FILKOM menggunakan bahasa pemrograman Java. Hasil survey menunjukkan sebanyak 76% dari 100 mahasiswa FILKOM lebih tertarik mempelajari materi selain mata kuliah pemrograman dasar. Keterbatasan sumber materi yang diperoleh dan penggunaan metode *Student Centered Learning* di FILKOM dinilai kurang membantu mahasiswa memahami materi pembelajaran. Oleh karena itu, telah dikembangkan sistem pembelajaran pemrograman java yang atraktif berbasis *website* menggunakan metode *gamification*. Akan tetapi, riset dari Studi Baidu menyatakan pengguna aplikasi *mobile* lebih besar yaitu 97% dibandingkan *browser* yang hanya 76%. Beberapa penelitian terkait *gamification* juga menyatakan bahwa arah penelitian selanjutnya adalah dari sisi mobilitas. Demi meningkatkan minat pengguna terhadap sistem pembelajaran pemrograman Java, perlu dikembangkan aplikasi pembelajaran pemrograman Java dalam versi *mobile* yang bertindak sebagai *client*, sehingga sistem yang sudah ada dapat bertindak serbagai *server*. Dalam penelitian ini, aplikasi dibangun dalam platform android *native*. Untuk melakukan pertukaran data dengan *server*, perlu dibangun *web service* yang digunakan sebagai *Application Programming Interface* (API). Hasil pengembangan aplikasi berbasis android diperoleh 19 fitur utama, sedangkan hasil pengembangan *web service* diperoleh 8 kelas baru pada sistem berbasis website. Pengujian unit dilakukan menggunakan metode *white-box* dan pengujian validasi dilakukan dengan metode *black-box*. Hasil dari pengujian unit dan pengujian validasi menghasilkan nilai 100%, yang berarti seluruh fitur bekerja sesuai dengan kebutuhan yang telah didefinisikan.

Kata kunci: pemrograman dasar, java, *gamification*, android, *web service*

ABSTRACT

Riski Pradana, Developement of Attractive Java Programming Learnig Application Based on Android

Advisor: Bayu Priyambadha, S.Kom, M.Kom dan Fajar Pradana, S.ST, M.Eng

In the Faculty of Computer Science (FILKOM) Brawijaya University, basic programming course must be taken by all students. It is use Java programming language. The survey shows that 76% of 100 FILKOM students are more interested in studying materials other than the basic programming courses. Limitations of material resources obtained and Student Centered Learning methods that used in FILKOM are considered less helpful for students to understand the learning materials. Therefore, it has developed an attractive java programming learning system based on website using gamification method. However, research from Studi Baidu states that users of mobile applications are 97%, greater than browsers that only 76%. Some studies related to gamification also states that the next research direction is from the side of mobility. In order to increase user interest in Java learning system learning programming, need to develop Java learning programming application in mobile version which acts as client, so that existing system can act as server. In this research, the application is built in native android platform. To exchange data with the server, it is necessary to build a web service that is used as Application Programming Interface (API). The results of the development of android-based applications obtained 19 main features, while the results of web service development obtained 8 new classes on the system-based website. Unit testing is done using white-box method and validation testing is done by black-box method. The results of unit testing and validation testing yield a 100% value, which means all features work in accordance with defined needs.

Keywords: basic programming, java, gamification, android, web service

DAFTAR ISI

| | |
|--|------|
| PENGESAHAN | ii |
| PERNYATAAN ORISINALITAS | iii |
| KATA PENGANTAR..... | iv |
| ABSTRAK..... | v |
| ABSTRACT | vi |
| DAFTAR ISI | vii |
| DAFTAR TABEL..... | x |
| DAFTAR GAMBAR..... | xiii |
| BAB 1 PENDAHULUAN..... | 1 |
| 1.1 Latar belakang..... | 1 |
| 1.2 Rumusan masalah..... | 2 |
| 1.3 Tujuan | 3 |
| 1.4 Manfaat..... | 3 |
| 1.5 Batasan masalah | 3 |
| 1.6 Sistematika pembahasan..... | 3 |
| BAB 2 LANDASAN KEPUSTAKAAN | 5 |
| 2.1 Tinjauan Pustaka | 5 |
| 2.2 Model Pengembangan Perangkat Lunak..... | 5 |
| 2.2.1 <i>Waterfall</i> | 6 |
| 2.3 <i>UML (Unified Modeling Language)</i> | 7 |
| 2.3.1 <i>Diagram Use Case</i> | 7 |
| 2.3.2 <i>Diagram Sequence</i> | 8 |
| 2.3.3 <i>Diagram Class</i> | 9 |
| 2.4 <i>Web Service</i> | 11 |
| 2.4.1 <i>Representational State Transfer</i> | 12 |
| 2.4.2 <i>Javascript Object Notation</i> | 12 |
| 2.5 <i>Android</i> | 12 |
| 2.5.1 <i>Activity</i> | 13 |
| 2.5.2 <i>Intent</i> | 14 |
| 2.5.3 <i>Fragment</i> | 14 |

| | |
|---|----|
| BAB 3 METODOLOGI | 17 |
| 3.1 Studi Literatur | 18 |
| 3.2 Analisis Kebutuhan | 18 |
| 3.3 Perancangan dan Implementasi | 18 |
| 3.4 Pengujian | 19 |
| 3.5 Kesimpulan dan Saran | 20 |
| BAB 4 Analisis kebutuhan..... | 21 |
| 4.1 Gambaran Umum Aplikasi | 21 |
| 4.1.1 Deskripsi Umum Aplikasi..... | 21 |
| 4.1.2 Lingkungan Sistem | 21 |
| 4.2 Proses Rekayasa Kebutuhan..... | 21 |
| 4.2.1 Elisitasi dan Spesifikasi Kebutuhan | 21 |
| 4.2.2 Pemodelan Diagram <i>Use case</i> | 28 |
| 4.2.3 Skenario <i>Use case</i> | 30 |
| BAB 5 PERANCANGAN DAN IMPLEMENTASI | 40 |
| 5.1 Perancangan | 40 |
| 5.1.1 Perancangan Sequence Diagram | 40 |
| 5.1.2 Perancangan <i>Class Diagram</i> | 45 |
| 5.1.3 Perancangan Komponen..... | 56 |
| 5.1.4 Perancangan Antarmuka..... | 58 |
| 5.1.5 Perancangan <i>Web Service</i> | 61 |
| 5.2 Implementasi Sistem | 62 |
| 5.2.1 Spesifikasi Sistem | 62 |
| 5.2.2 Implementasi Kode Program | 63 |
| 5.2.3 Implementasi Antar Muka | 66 |
| 5.2.4 Implementasi <i>Web Service</i> | 70 |
| BAB 6 PENGUJIAN | 72 |
| 6.1 Pengujian Unit..... | 72 |
| 6.1.1 Pengujian Unit Kelas <i>TakeExerciseActivity</i> untuk operasi <i>btCompile()</i> | 72 |
| 6.1.2 Pengujian Unit Kelas <i>DialogFragmentChallengeConfirmation</i> untuk operasi <i>onCreateDialog()</i> | 74 |

| | |
|---|----|
| 6.1.3 Pengujian Unit Kelas DialogFragmentAddFriendController untuk operasi onCreateDialog() | 75 |
| 6.2 Pengujian Validasi | 77 |
| 6.2.1 Pengujian Validasi Fungsi Register | 77 |
| 6.2.2 Pengujian Validasi Fungsi Login | 78 |
| 6.2.3 Pengujian Validasi Fungsi Melihat Profil | 79 |
| 6.2.4 Pengujian Validasi Fungsi Melakukan Perubahan Profil | 79 |
| 6.2.5 Pengujian Validasi Fungsi Melihat Detail Latihan | 80 |
| 6.2.6 Pengujian Validasi Fungsi Melihat Detail Tantangan | 80 |
| 6.2.7 Pengujian Validasi Fungsi Melihat Daftar Teman | 80 |
| 6.2.8 Pengujian Validasi Fungsi Mengirim Permintaan Pertemanan .. | 80 |
| 6.2.9 Pengujian Validasi Fungsi Melihat Daftar Permintaan Pertemanan | 81 |
| 6.2.10 Pengujian Validasi Fungsi Menerima Permintaan Pertemanan | 81 |
| 6.2.11 Pengujian Validasi Fungsi Mengajukan Tantangan..... | 82 |
| 6.2.12 Pengujian Validasi Fungsi Melihat Materi Latihan..... | 82 |
| 6.2.13 Pengujian Validasi Fungsi Mengerjakan Latihan..... | 83 |
| 6.2.14 Pengujian Validasi Fungsi Mengerjakan Tantangan | 83 |
| 6.2.15 Pengujian Validasi Fungsi Melakukan Kompilasi Kode Java | 84 |
| 6.2.16 Pengujian Validasi Fungsi Melakukan Percobaan Kode Java.... | 84 |
| 6.2.17 Pengujian Validasi Fungsi Menjawab Tantangan..... | 84 |
| 6.2.18 Pengujian Validasi Fungsi Menolak Permintaan Pertemanan .. | 85 |
| 6.2.19 Pengujian Validasi Fungsi <i>Logout</i> | 85 |
| 6.2.20 Pengujian Validasi Non Fungsional <i>Compatibility</i> | 85 |
| BAB 7 Penutup | 87 |
| 7.1 Kesimpulan..... | 87 |
| 7.2 Saran | 87 |
| DAFTAR PUSTAKA..... | 88 |

DAFTAR TABEL

| | |
|--|----|
| Tabel 4.1 Temuan Kebutuhan | 22 |
| Tabel 4.2 Identifikasi Aktor | 22 |
| Tabel 4.3 Penjelasan Aturan Penomoran Kebutuhan | 23 |
| Tabel 4.4 Daftar Kebutuhan Fungsional Pengguna | 23 |
| Tabel 4.5 Daftar Kebutuhan Fungsional Member | 23 |
| Tabel 4.6 Daftar Kebutuhan Non Fungsional | 27 |
| Tabel 4.7 Skenario <i>Use case</i> Register | 31 |
| Tabel 4.8 Skenario <i>Use case</i> Login | 31 |
| Tabel 4.9 Skenario <i>Use case</i> Melihat Profil | 32 |
| Tabel 4.10 Skenario <i>Use case</i> Melakukan Perubahan Profil | 32 |
| Tabel 4.11 Skenario <i>Use case</i> Melihat Detail Latihan | 33 |
| Tabel 4.12 Skenario <i>Use case</i> Melihat Detail Tantangan | 33 |
| Tabel 4.13 Skenario <i>Use case</i> Melihat Daftar Teman | 33 |
| Tabel 4.14 Skenario <i>Use case</i> Mengirim Permintaan Pertemanan | 34 |
| Tabel 4.15 Skenario <i>Use case</i> Melihat Daftar Permintaan Pertemanan | 34 |
| Tabel 4.16 Skenario <i>Use case</i> Menerima Permintaan Pertemanan | 35 |
| Tabel 4.17 Skenario <i>Use case</i> Mengajukan Tantangan | 35 |
| Tabel 4.18 Skenario <i>Use case</i> Melihat Materi Latihan | 36 |
| Tabel 4.19 Skenario <i>Use case</i> Mengerjakan Latihan | 36 |
| Tabel 4.20 Skenario <i>Use case</i> Mengerjakan Tantangan | 37 |
| Tabel 4.21 Skenario <i>Use case</i> Melakukan Kompilasi Kode Java | 37 |
| Tabel 4.22 Skenario <i>Use case</i> Melakukan Percobaan Kode Java | 38 |
| Tabel 4.23 Skenario <i>Use case</i> Menjawab Tantangan | 38 |
| Tabel 4.24 Skenario <i>Use case</i> Menolak Permintaan Petemanan | 38 |
| Tabel 4.25 Skenario <i>Use case</i> Logout | 39 |
| Tabel 5.1 Algoritme Melakukan Kompilasi Kode Java | 57 |
| Tabel 5.2 Algoritme Mengajukan Tantangan | 57 |
| Tabel 5.3 Algoritme Mengirim Permintaan Pertemanan | 58 |
| Tabel 5.4 Spesifikasi Perangkat Keras | 63 |
| Tabel 5.5 Spesifikasi Perangkat Lunak | 63 |

| | |
|---|-----|
| Tabel 5.6 Kode Program Melakukan Kompilasi Kode Java | 64 |
| Tabel 5.7 Kode Program Mengajukan Tantangan..... | 64 |
| Tabel 5.8 Mengirimkan Permintaan Pertemanan..... | 65 |
| Tabel 6.1 <i>Pseudocode</i> Pengujian Unit Operasi btCompile() | 72 |
| Tabel 6.2 <i>Pseudocode</i> Pengujian Unit Operasi onCreateDialog() | 74 |
| Tabel 6.3 <i>Pseudocode</i> Pengujian Unit Operasi btSubmitChallenge() | 75 |
| Tabel 6.4 Pengujian validasi fungsi regitser | 77 |
| Tabel 6.5 Pengujian validasi fungsi register alternatif 1 | 77 |
| Tabel 6.6 Pengujian validasi fungsi register alternatif 2 | 78 |
| Tabel 6.7 Pengujian validasi fungsi login | 78 |
| Tabel 6.8 Pengujian validasi login alternatif 1 | 78 |
| Tabel 6.9 Pengujian validasi login alternatif 2 | 79 |
| Tabel 6.10 Pengujian validasi fungsi melihat profil | 79 |
| Tabel 6.11 Pengujian validasi melakukan perubahan profil | 79 |
| Tabel 6.12 Pengujian validasi fungsi melihat detail latihan..... | 80 |
| Tabel 6.13 Pengujian validasi fungsi melihat detail tantangan | 80 |
| Tabel 6.14 Pengujian validasi fungsi melihat daftar teman..... | 80 |
| Tabel 6.15 Pengujian validasi fungsi mengirim permintaan pertemanan | 80 |
| Tabel 6.16 Pengujian validasi mengirim permintaan pertemanan alternatif 1.... | 81 |
| Tabel 6.17 Pengujian validasi fungsi melihat permintaan pertemanan | 81 |
| Tabel 6.18 Pengujian validasi fungsi menerima permintaan pertemanan | 81 |
| Tabel 6.19 Pengujian validasi fungsi mengajukan tantangan..... | 82 |
| Tabel 6.20 Pengujian validasi fungsi mengajukan tantangan alternatif 1 | 82 |
| Tabel 6.21 Pengujian validasi fungsi melihat materi latihan | 82 |
| Tabel 6.22 Pengujian validasi fungsi mengerjakan latihan | 83 |
| Tabel 6.23 Pengujian validasi fungsi mengerjakan tantangan..... | 83 |
| Tabel 6.24 Pengujian validasi fungsi melakukan kompilasi kode Java | 84 |
| Tabel 6.25 Pengujian validasi fungsi melakukan kompilasi kode Java alternatif | 184 |
| Tabel 6.26 Pengujian validasi fungsi melakukakan percobaan kode Java | 84 |
| Tabel 6.27 Pengujian validasi fungsi menjawab tantangan..... | 84 |
| Tabel 6.28 Pengujian validasi fungsi menolak permintaan pertemanan | 85 |
| Tabel 6.29 Pengujian validasi fungsi <i>logout</i> | 85 |

Tabel 6.30 Pengujian validasi *compatibility* pada sistem operasi android versi 8 85

Tabel 6.31 Pengujian validasi *compatibility* pada sistem operasi android versi 5 86



DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2.1 Model <i>Waterfall</i> | 6 |
| Gambar 2.2 Contoh <i>Use Case Diagram</i> | 8 |
| Gambar 2.3 Contoh Diagram <i>Sequence</i> | 9 |
| Gambar 2.4 Contoh <i>Class Diagram</i> | 10 |
| Gambar 2.5 Contoh Hubungan Generalisasi..... | 10 |
| Gambar 2.6 Contoh Hubungan Agregasi | 11 |
| Gambar 2.7 Contoh JSON..... | 12 |
| Gambar 2.8 <i>Activity lifecycle</i> | 13 |
| Gambar 2.9 Daur hidup <i>fragment</i> | 15 |
| Gambar 3.1 Digram Alir Metodologi Penelitian..... | 17 |
| Gambar 4.1 Aturan Penomoran Kebutuhan | 23 |
| Gambar 4.2 Diagram <i>Use case</i> | 29 |
| Gambar 5.1 <i>Sequence Diagram</i> Melakukan Kompilasi Kode Java | 41 |
| Gambar 5.2 <i>Sequence Diagram</i> Mengajukan Tantangan | 43 |
| Gambar 5.3 <i>Sequence Diagram</i> Mengirim Permintaan Pertemanan | 45 |
| Gambar 5.4 <i>Class Diagram</i> | 46 |
| Gambar 5.5 Informasi <i>Class Activity</i> | 48 |
| Gambar 5.6 Informasi <i>Class Fragment</i> | 50 |
| Gambar 5.7 Informasi <i>Class Controller</i> | 51 |
| Gambar 5.8 Informasi <i>Class Model</i> | 52 |
| Gambar 5.9 Informasi <i>Class Model</i> | 53 |
| Gambar 5.10 Informasi <i>Class Response</i> | 54 |
| Gambar 5.11 Informasi <i>Class Adapter</i> | 55 |
| Gambar 5.12 Informasi <i>Class API</i> | 56 |
| Gambar 5.13 Rancangan Antarmuka Halaman Mengerjakan Latihan | 59 |
| Gambar 5.14 Rancangan Antarmuka Halaman Mengajukan Tantangan | 60 |
| Gambar 5.15 Rancangan Antarmuka Halaman Menemukan Teman Baru..... | 61 |
| Gambar 5.16 Rancangan <i>Web Service</i> | 62 |
| Gambar 5.17 Implementasi Antarmuka Halaman Mengerjakan Latihan | 67 |
| Gambar 5.18 Implementasi Antarmuka Halaman Mengajukan Tantangan | 68 |

| | |
|---|----|
| Gambar 5.19 Implementasi Antarmuka Halaman Cari Teman | 69 |
| Gambar 6.1 <i>Flowgraph</i> fungsi <i>btCompile</i> | 73 |
| Gambar 6.2 <i>Flowgraph</i> fungsi <i>onCreateDialog</i> | 74 |
| Gambar 6.3 <i>Flowgraph</i> fungsi <i>btSubmitChallenge</i> | 76 |



BAB 1 PENDAHULUAN

1.1 Latar belakang

Di Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya, pemrograman dasar menjadi mata kuliah yang wajib ditempuh oleh seluruh mahasiswa. Mata kuliah pemrograman dasar dilakukan dengan menggunakan bahasa pemrograman Java. Berdasarkan survey yang dilakukan pada 22 Februari 2017 di FILKOM dengan responden sebanyak 100 mahasiswa dari angkatan 2013-2016 dari program studi Teknik Informatika, Sistem Informasi dan Sistem Komputer didapatkan hasil bahwa 76% dari responden lebih tertarik mempelajari materi selain mata kuliah pemrograman dasar (Bastari, et al., 2017). Salah satu faktor yang mempengaruhi kurangnya minat mahasiswa terhadap mata kuliah pemrograman dasar adalah keterbatasan sumber materi yang diperoleh yaitu hanya terbatas pada penyampaian dosen di kelas dan asisten praktikum. Saat ini sistem pembelajaran di FILKOM menggunakan metode *Student Centered Learning*, yaitu metode yang menempatkan pelajar sebagai peserta didik yang aktif serta mandiri sehingga pelajar bertanggung jawab penuh atas proses pembelajarannya. Berdasarkan hasil survey berupa kuisioner menyatakan bahwa metode pembelajaran tersebut dinilai kurang membantu mahasiswa memahami materi pembelajaran. Salah satu contoh kasus yang terjadi adalah saat pemberian tugas oleh dosen dan asisten yang bersifat kelompok menyebabkan mahasiswa yang kurang memahami materi bergantung kepada teman kelompoknya yang sudah menguasai materi. Untuk mengatasi permasalahan tersebut, telah dibangun sistem pembelajaran pemrograman Java menggunakan metode *gamification* pada penelitian yang berjudul "*Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*" oleh Bastari, et al. (2017).

Saat ini *gamification* berfokus pada penerapan fitur *point*, *level*, *leaderboard* dan *badges* (Pedreira, et al., 2014). Seperti halnya pada sistem pembelajaran pemrograman Java yang atraktif berbasis *website*, penerapan metode *gamification* dapat dilihat pada fitur menantang pengguna lain (*challenge*), penghargaan bagi pengguna dengan pencapaian tertentu (*achievement*), *level* dan *experience point*. Sedangkan, salah satu arah dalam penelitian masa mendatang adalah pada sisi mobilitas (Pedreira, et al., 2014). Hal ini dilakukan agar pengguna dengan mobilitas tinggi dapat mengakses sistem yang dikembangkan secara mudah. Salah satu cara agar sistem dapat mendukung mobilitas adalah dengan menggunakan perangkat *mobile* sebagai platform pengembangannya, yaitu dalam bentuk aplikasi *mobile*. Aplikasi *mobile* mendukung pengguna dengan mobilitas tinggi serta memberikan kesan mudah, nyaman dan praktis bagi pengguna. Meskipun saat ini *website* sudah mendukung untuk bisa diakses melalui perangkat *mobile* yaitu menggunakan teknologi *mobile web*, terdapat beberapa hal yang membuat *aplikasi mobile* lebih diminati dibandingkan *mobile web*. Salah satu diantaranya adalah performa dari *mobile*

web relatif lambat dan tidak semua perangkat memiliki fitur tertentu yang ada pada *mobile web*. Berbeda dengan aplikasi *mobile* yang memiliki performa lebih cepat, lebih menarik dari segi visual dan mendukung akses penuh terhadap perangkat seperti kamera, *gesture*, pemberitahuan, *speaker* dan fitur lainnya.

Mengacu pada riset Gfk Indonesia bertajuk *Mobile Aps Market Study* Indonesia yang dilakukan Studi Baidu terhadap lebih dari 2.200 orang responden di Jakarta, Bandung, Bogor, Tangerang, Bekasi, Semarang dan Surabaya, menyatakan bahwa pengguna aplikasi *mobile* lebih besar yaitu 97% dibandingkan pengguna yang mengakses *browser* yang hanya 76%. Hal ini menunjukkan bahwa aplikasi *mobile* lebih populer dan lebih diminati oleh pengguna. Head of Marketing Baidu Indonesia Iwan Setiawan mengatakan bahwa rata-rata pengguna *mobile* di Indonesia meluangkan waktu 60 menit per hari untuk berinteraksi dengan aplikasi *mobile* yang telah diunduhnya ke *smartphone* atau tablet. Dalam survey tersebut juga mencatat beberapa aplikasi yang paling populer di Indonesia, diantaranya adalah *game* dengan jumlah unduhan sebanyak 38%, olah pesan sebanyak 27% dan media sosial sebanyak 19%. Dari data tersebut diketahui bahwa aplikasi jenis *game* memiliki jumlah unduhan tertinggi. Berdasarkan data tersebut bisa disimpulkan bahwa pengguna aplikasi *mobile* dengan jenis *game* adalah jenis aplikasi yang paling banyak menarik minat pengguna.

Berdasarkan permasalahan yang sudah dipaparkan terkait pengembangan *gamification* dari sisi mobilitas dan data terkait pengguna aplikasi *mobile*, solusi yang ditawarkan dalam penelitian ini adalah dengan mengembangkan aplikasi pembelajaran pemrograman Java yang disajikan dalam bentuk permainan simulasi yang atraktif dalam platform android. Aplikasi yang dibangun akan bertindak sebagai *client*. Sebagai *client*, aplikasi hanya bertugas mengirimkan atau menerima data, sedangkan seluruh proses komputasi dilakukan pada sisi *server*, yaitu sistem pembelajaran pemrograman Java berbasis *website* yang sudah ada. Sebagai sarana komunikasi antara *client* dan *server*, solusi yang ditawarkan berikutnya yaitu pembuatan *web service* yang berfungsi sebagai *Application Programming Interface* (API). Dengan dibangunnya aplikasi ini diharapkan dapat minat pengguna untuk belajar pemrograman Java.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka penelitian ini membahas beberapa permasalahan sebagai berikut:

1. Bagaimana analisis kebutuhan, perancangan, implementasi dan pengujian aplikasi pembelajaran pemrograman Java berbasis android?
2. Bagaimana pengembangan *web service* pada sistem pembelajaran pemrograman Java?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan aplikasi pembelajaran pemrograman Java yang atraktif dalam bentuk permainan simulasi berbasis android.
2. Meningkatkan minat pengguna dalam belajar bahasa pemrograman Java sehingga dapat menunjang proses perkuliahan mahasiswa khususnya pada mata kuliah pemrograman dasar dan juga seluruh pengguna secara umum.

1.4 Manfaat

Pelaksanaan penelitian ini diharapkan dapat menghasilkan manfaat sebagai berikut:

1. Bagi Penulis:

Penulis dapat melakukan implementasi dari ilmu yang telah diperoleh dalam melakukan perancangan serta pembangunan sebuah sistem pembelajaran pemrograman Java.

2. Bagi Pengguna:

Menjadi sarana pembelajaran baru yang dapat meningkatkan minat serta pengetahuan terhadap pemrograman Java.

3. Bagi Pembaca atau pihak lain:

Menjadi bahan acuan bagi para pembaca atau pihak lain untuk pengkajian topik yang berkaitan dengan permasalahan yang ada dalam penelitian ini.

1.5 Batasan masalah

Dalam penelitian ini penulis memberikan batasan-batasan permasalahan yang dibahas sebagai berikut:

1. Aplikasi pembelajaran pemrograman Java dibangun menggunakan platform android *native*.
2. Pembuatan *web service* mencakup seluruh fungsi utama dari sistem pembelajaran pemrograman Java.

1.6 Sistematika pembahasan

BAB I PENDAHULUAN

Pada bab ini penulis membahas tentang latar belakang permasalahan yang menjadi alasan dilakukannya penelitian, perumusan masalah, tujuan dilakukannya penelitian, manfaat dilakukannya penelitian, batasan masalah yang menjadi batasan dalam penelitian yang dilakukan dan sistematika dalam menuliskan penelitian.

BAB II LANDASAN KEPUSTAKAAN

Pada bab ini membahas sub bab terkait teori penelitian, antara lain tinjauan pustaka, model pengembangan perangkat lunak, teori perancangan dan pengembangan aplikasi, teori pengembangan aplikasi android dan teori tentang *web service*.

BAB III METODOLOGI

Pada bab ini membahas metode penelitian yang digunakan, diantaranya studi literatur, analisis kebutuhan aplikasi, perancangan aplikasi, implementasi aplikasi serta pengujian aplikasi.

BAB IV ANALISIS KEBUTUHAN

Pada bab ini membahas proses analisis kebutuhan dari aplikasi yang dikembangkan untuk menjadi dasar dalam tahap perancangan.

BAB V PERANCANGAN DAN IMPLEMENTASI

Pada bab ini membahas proses perancangan aplikasi pembelajaran pemrograman Java berbasis android serta proses implementasi aplikasi berdasarkan hasil perancangan yang dilakukan.

BAB VI PENGUJIAN

Pada bab ini membahas proses pengujian aplikasi yang telah dibangun serta menjelaskan hasil dari pengujian aplikasi yang dilakukan.

BAB VII PENUTUP

Pada bab ini membahas kesimpulan yang diperoleh dari penelitian yang dilakukan serta memaparkan saran sebagai dasar untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas tentang teori yang menjadi dasar dalam penelitian yang dilakukan, diantaranya terkait model pengembangan perangkat lunak, teori seputar Rekayasa Perangkat Lunak, teori terkait *web service*, teori pengembangan aplikasi android. Pada bagian Tinjauan Pustaka membahas mengenai penelitian sebelumnya yang terkait, yaitu "*Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*". Hasil penelitian tersebut menjadi dasar dalam penelitian ini.

2.1 Tinjauan Pustaka

Penelitian ini dilakukan berdasarkan penelitian sebelumnya yang berjudul "*Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*" yang dilakukan oleh Dhanuari Indra Bastari pada tahun 2017. Pada penelitian tersebut, permasalahan yang terjadi adalah kurangnya minat mahasiswa FILKOM terhadap mata kuliah pemrograman dasar. Untuk mengatasi permasalahan tersebut dikembangkan sistem pembelajaran pemrograman Java berbasis *website* menggunakan metode *gamification*. Metode *gamification* adalah sebuah metode yang digunakan untuk membuat aplikasi *non-game* memiliki *behavior* layaknya *game*. Metode *gamification* dipilih untuk membuat sistem lebih menarik minat pengguna. Beberapa karakteristik dari metode *gamification* adalah adanya *point*, *level*, *leaderboard* dan *badges* pada aplikasi yang dikembangkan.

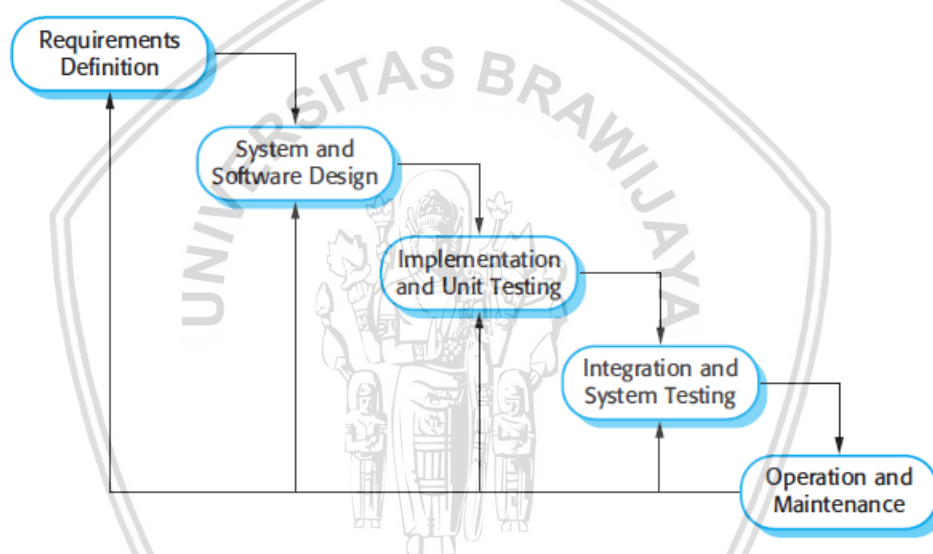
Untuk mendukung mobilitas, kenyamanan dan kemudahan saat diakses oleh pengguna, maka sistem pembelajaran pemrograman Java ini harus mendukung untuk bisa digunakan dalam perangkat *mobile* dalam bentuk *mobile apps*. Aplikasi pada perangkat *mobile* bertindak sebagai *client*. Untuk keperluan komunikasi melakukan pertukaran data dengan sistem pembelajaran pemrograman Java berbasis *website* sebagai *server*, maka dalam penelitian ini dibangun *web service* sebagai *Application Programming Interface* (API). API digunakan sebagai *interface* atau jembatan antara aplikasi *mobile* sebagai *client* yang menerima atau mengirim data dengan sistem berbasis *webiste* yang sudah ada sebagai *server* yang melakukan seluruh proses komputasi.

2.2 Model Pengembangan Perangkat Lunak

Model dalam proses pengembangan perangkat lunak dapat diartikan sebagai sebuah representasi sederhana dari sebuah proses pengembangan perangkat lunak (Sommerville, 2011). Terdapat beberapa model pengembangan perangkat lunak, salah satunya adalah model yang dipakai dalam penelitian ini yaitu *waterfall*. Selain model *waterfall* ada juga model lain diantaranya *incremental*, *prototyping*, *spiral*, *rapid development* dan *extreme programming*.

2.2.1 Waterfall

Model *waterfall* mengambil aktivitas proses dasar dari spesifikasi, *development*, validasi, dan evolusi dan merepresentasikannya ke dalam beberapa fase proses diantaranya spesifikasi kebutuhan, perancangan, implementasi, pengujian dan sebagainya (Sommerville, 2011). Pada model *waterfall*, pengembangan dilakukan tahap per tahap secara berurutan atau sekuensial. Tahap-tahap pada model *waterfall* direpresentasikan menyerupai air terjun, dimana tahap pertama terletak di posisi paling atas diikuti tahap berikutnya di bawah dan seterusnya. Hasil dari sebuah tahap akan dijadikan sebagai dasar atau input pada tahap berikutnya. Dengan demikian, proses pengerjaan sebuah tahap tidak dapat dilakukan sebelum tahap sebelumnya benar-benar selesai dilakukan. Beberapa fase proses pada model *waterfall* dapat dilihat pada Gambar 2.1 berikut.



Gambar 2.1 Model Waterfall

(Sumber: Sommerville, 2011)

Berikut penjelasan mengenai tahapan-tahapan pada model *waterfall* menurut Sommerville (2011).

1. *Requirement analysis and definition*, yaitu menetapkan layanan sistem, batasan sistem dan tujuan sistem dengan pengguna dan mendefinisikan temuan secara rinci untuk dijadikan sebagai spesifikasi sistem.
2. *System and software design*, yaitu menggunakan temuan-temuan kebutuhan untuk membangun sebuah arsitektur sistem secara keseluruhan dengan cara mengidentifikasi dan mengGambarkan abstraksi sistem beserta hubungan-hubungan.
3. *Implementation and unit testing*, yaitu merealisasikan rancangan sistem dalam sebuah program atau unit program. Selanjutnya, pengujian unit

dilakukan untuk memastikan setiap unit dari sistem telah sesuai dengan spesifikasinya.

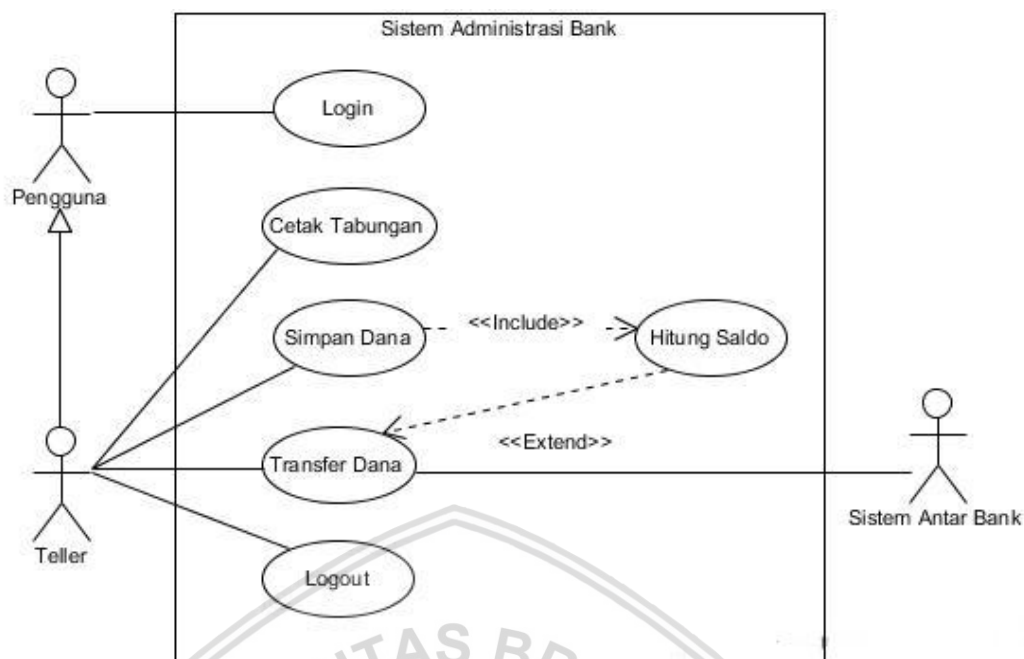
4. *Integration and system testing*, yaitu setiap unit program yang telah diintegrasikan kemudian diuji sebagai sebuah sistem yang utuh untuk memastikan bahwa seluruh kebutuhan sistem telah terpenuhi. Selanjutnya, sistem siap di serahkan kepada klien.
5. *Operation and maintenance*, yaitu tahap menginstal sistem dan membuat petunjuk penggunaan. Selanjutnya, dilakukan pemerliharaan sistem yang melibatkan perbaikan error, peningkatan implementasi unit sistem dan meningkatkan layanan sistem.

2.3 UML (Unified Modeling Language)

Unified Modelling Language (UML) digunakan sebagai standar dalam pemodelan perangkat lunak untuk mengGambarkan, membangun serta mendokumentasikan artifak dalam proses pengembangan perangkat lunak (Pressman, 2010). Tujuan dari UML adalah mengGambarkan sebuah mekanisme pemodelan yang mudah dipahami oleh *stakeholder*. Jenis-jenis diagram UML diantaranya adalah *use case diagram*, *class diagram*, dan *sequence diagram*.

2.3.1 Diagram Use Case

Use case diagram adalah sebuah diagram yang menjelaskan perilaku sistem yang tampak dari luar. *Use case diagram* digunakan untuk memodelkan interaksi antara sistem dengan aktor (pengguna atau sistem lain) (Sommerville, 2011). *Use case* menyediakan fungsi-fungsi yang harus dipenuhi sistem sesuai dengan aktornya. Elemen utama dari use case adalah sebuah *actor* (orang atau sistem lain) dan *use case* yang merepresentasikan sebagai sebuah fungsionalitas sistem. Langkah-langkah pembuatan *use case* adalah dengan mengidentifikasi aktor. Selanjutnya dilakukan identifikasi *use case* per aktor. Contoh use case diagram dapat dilihat pada Gambar 2.2 berikut.



Gambar 2.2 Contoh Use Case Diagram

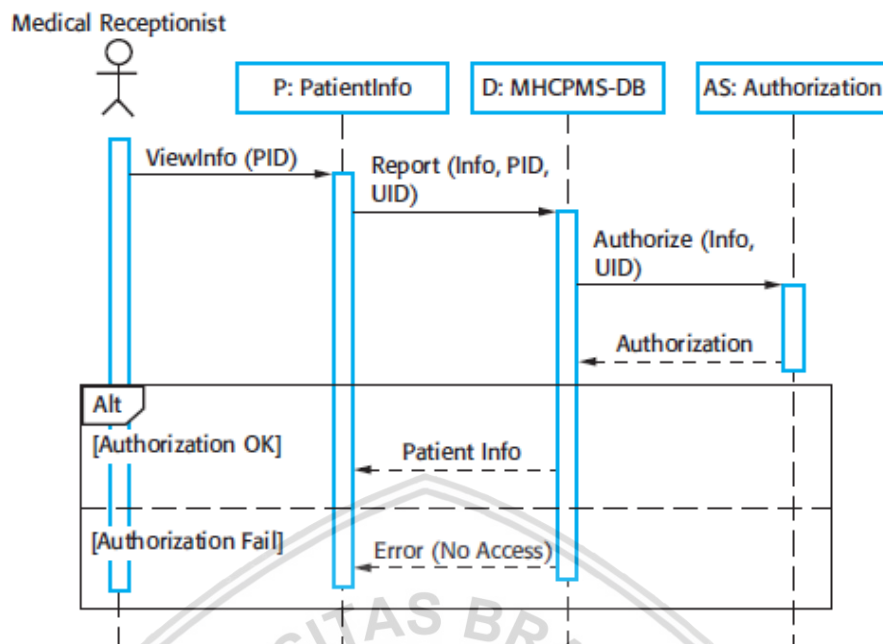
(Sumber: Kurniawan, 2018)

Menurut Kurniawan (2018), notasi-notasi yang ada pada *use case diagram* antara lain:

1. Aktor, yang dinotasikan menggunakan simbol orang-orangan (*stick-man*) dengan keterangan nama berupa kata benda yang menyatakan peran/sistem.
2. Use case, yang dinotasikan menggunakan simbol elips dengan keterangan kata kerja aktif di dalamnya yang menyatakan aktivitas dari perspektif aktor.
3. Relasi, merupakan hubungan yang terjadi baik antar aktor maupun antar *use case*. Sebuah *use case* utama (*base use case*) dapat memiliki hubungan dengan satu atau lebih *use case* yang lain (*supplier use case*). Hubungan yang terjadi bisa dalam bentuk *extends*, yang menyatakan bahwa fungsionalitas *base use case* bisa diperluas oleh *supplier use case*, jika dibutuhkan ataupun dalam bentuk *include*, yang menyatakan fungsionalitas dari *base use case* hanya bisa dipenuhi dengan bantuan *supplier use case*. Selain itu, terdapat relasi yang bernama inheritance yang biasanya digunakan untuk menggambarkan hubungan antara aktor induk dengan aktor turunan.

2.3.2 Diagram Sequence

Dalam UML, *sequence diagram* digunakan untuk memodelkan interaksi antara aktor dengan objek dan interaksi antar objek dengan objek lainnya dalam satu sistem (Sommerville, 2011). Sequence diagram menunjukkan kumpulan objek dan pesan yang dikirim atau diterima oleh objek-objek tersebut dengan memperhatikan urutan waktu. Contoh sequence diagram dapat dilihat pada Gambar 2.3 berikut.



Gambar 2.3 Contoh Diagram Sequence

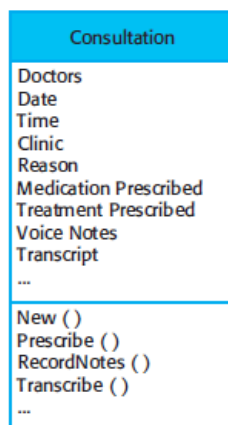
(Sumber: Sommerville, 2011)

Berdasarkan Gambar 2.3 diatas, beberapa notasi yang terdapat di dalam sequence diagram antara lain:

- Actor*, yang direpresentasikan menggunakan Gambar orang-orangan (*stick-man*).
- Object*, yang direpresentasikan menggunakan kotak dan di dalamnya berisi nama objek.
- Dashed line*(atau disebut dengan *lifeline*), yang dipresentasikan dalam garis vertikal pada masing-masing objek yang menunjukkan keberadaan sebuah objek dalam waktu tertentu.
- Message*, yang direpresentasikan dalam garis horizontal yang dilewatkan dari objek menuju objek yang lain yang semakin menurun seiring dengan waktu pemanggilan objek.

2.3.3 Diaram Class

Class Diagram digunakan untuk mengembagkan sistem dengan model *object-oriented* untuk menunjukkan kelas-kelas yang ada pada sebuah sistem dan hubungan-hubungan yang terjadi antar kelas-kelas tersebut (Sommerville, 2011). Contoh class diagram dapat dilihat pada Gambar2.4 berikut.



Gambar 2.4 Contoh Class Diagram

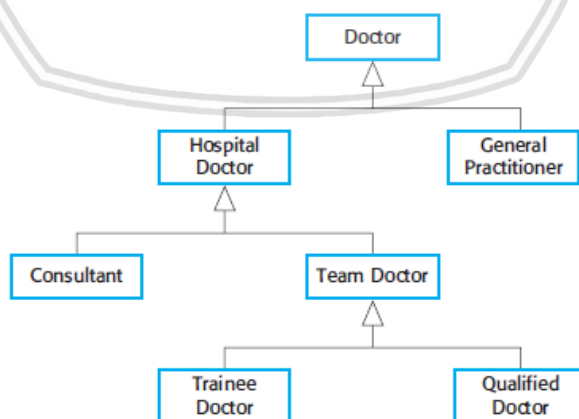
(Sumber: Sommerville, 2011)

Berdasarkan Gambar 2.4 diatas, dalam sebuah *class diagram* terdapat beberapa bagian, diantaranya:

1. Nama kelas di bagian teratas kelas.
2. Atribut kelas di bagian tengah. Atribut harus memiliki nama atribut berserta tipe atribut (opsional).
3. Operasi(atau jika pada Java atau bahas pemrograman berorientasi objek lainnya dinamakan *method*), yang ditunjukkan pada bagian bawah kelas.

Dalam *class diagram* terdapat beberapa jenis relasi, yaitu yang menyatakan hubungan antara suatu kelas dengan kelas lainnya. Beberapa jenis relasi diantaranya:

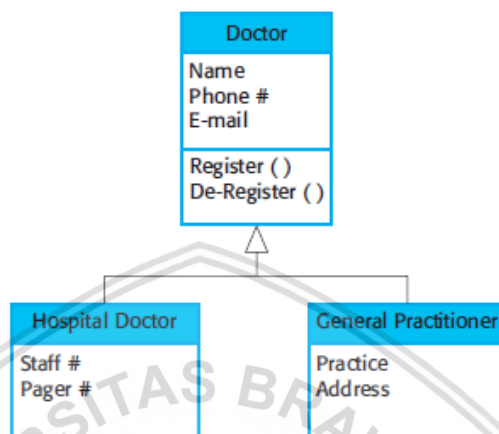
1. Generalisasi, yaitu hubungan yang menyatakan bahwa suatu kelas yang lebih umum memiliki kelas turunan dibawahnya yang lebih khusus. Relasi ini biasanya dinyatakan dengan hubungan "is-a" atau *inheritance*. Contoh generalisasi dapat dilihat pada Gambar 2.5 berikut.



Gambar 2.5 Contoh Hubungan Generalisasi

(Sumber: Sommerville, 2011)

2. Asosiasi, yaitu hubungan umum antara 2 kelas yang dinyatakan sebuah garis lurus dari satu kelas menuju kelas lainnya.
3. Agregasi, yaitu hubungan yang menyatakan bahwa satu objek (yang dinyatakan dalam sebuah kelas) terdiri dari objek yang lain, atau biasa disebut “part-of”. Contoh hubungan agregasi dapat dilihat pada Gambar 2.6 berikut.



Gambar 2.6 Contoh Hubungan Agregasi
(Sumber: Sommerville, 2011)

4. Komposisi, yaitu hubungan yang menyatakan bahwa sebuah kelas tidak dapat berdiri sendiri dan memiliki ketergantungan dengan kelas yang lain.

2.4 Web Service

Web service adalah aplikasi mini yang mengekspose suatu fungsi yang dapat diakses menggunakan standar teknologi *web* dan sesuai dengan standar *web service* (Mutiawani, et al., 2017). *Web service* dirancang untuk mendukung interoperabilitas interaksi antar mesin melalui jaringan. Sebuah *web service* biasanya hanya berisi fungsi-fungsi tertentu yang dibutuhkan oleh sistem lain. Tujuan dibuatnya *web service* adalah untuk membuat sebuah fungsi bisa digunakan kembali dan dibagikan kepada aplikasi klien meskipun menggunakan platform yang berbeda-beda seperti *mobile*, *desktop* ataupun aplikasi *web*. *Web service* dikembangkan ke dalam platform apapun serta menggunakan bahasa apapun. Seluruh *web service* menggunakan HTTP (Hypertext Transfer Protocol) dan format pertukaran data standar dalam bentuk *XML (Extensible Markup Language)*, *JSON (Javascript Object Notation)* atau media lain. Ada 2 jenis *web service* yaitu *SOAP (Simple Object Access Protocol)* dan *REST (Representational State Transfer)*.

2.4.1 Representational State Transfer

Representational State Transfer (REST) adalah seperangkat batasan-batasan yang akan membuat gaya arsitektural sistem (Mutiawani, et al., 2017). Batasan-batasan tersebut diantaranya harus berupa *client-server system*, *stateless*, dapat diakses secara seragam, dalam *layer-layer*, mendukung sistem *caching* dan menyediakan kode sesuai dengan permintaan atau kebutuhan. Dalam sistem RESTful, klien dan *server* hanya berinteraksi dengan saling mengirim pesan yang mengikuti protokol yang telah ditentukan (Richardson & Amundsend, 2013). Dalam dunia web *Application Programming Interface* (API), protokol yang dimaksud adalah HTTP. Standar dalam HTTP mendefinisikan 8 jenis pesan yang berbeda, akan tetapi umumnya hanya 4 yang sering digunakan yaitu diantaranya:

1. *GET*, digunakan untuk mendapatkan sebuah representasi dari *resource*.
2. *DELETE*, digunakan untuk menghapus *resource*.
3. *POST*, digunakan untuk membuat *resource* baru dibawah *resource* yang lain, berdasarkan representasi yang diberikan.
4. *PUT*, digunakan untuk mengubah *state* dari sebuah *resource* dengan *resource* yang didefinisikan pada representasi yang diberikan.

2.4.2 Javascript Object Notation

Javascript Object Notation (JSON) adalah sebuah standar untuk merepresentasikan struktur data sederhana dalam *plain text* (Richardson & Amundsend, 2013). Sebuah objek JSON dibentuk dalam format *key-value* yang dibungkus dalam kurung kurawal. Contoh JSON dapat dilihat pada Gambar 2.7 berikut

```
{"key": "value"}
```

Gambar 2.7 Contoh JSON

(Sumber : Richardson & Amundsend, 2013)

Gambar 2.7 diatas menggambarkan *JSON* objek yang hanya memiliki satu data saja, yaitu hanya memiliki satu pasang *key-value*. Jika memiliki lebih dari satu maka setelah pasangan *key-value* ditambahkan separator berupa tanda koma (,). Dalam level yang lebih lanjut, *JSON* dapat berisi beberapa objek data, dimana pada satu objek data berisi beberapa pasangan *key-value*.

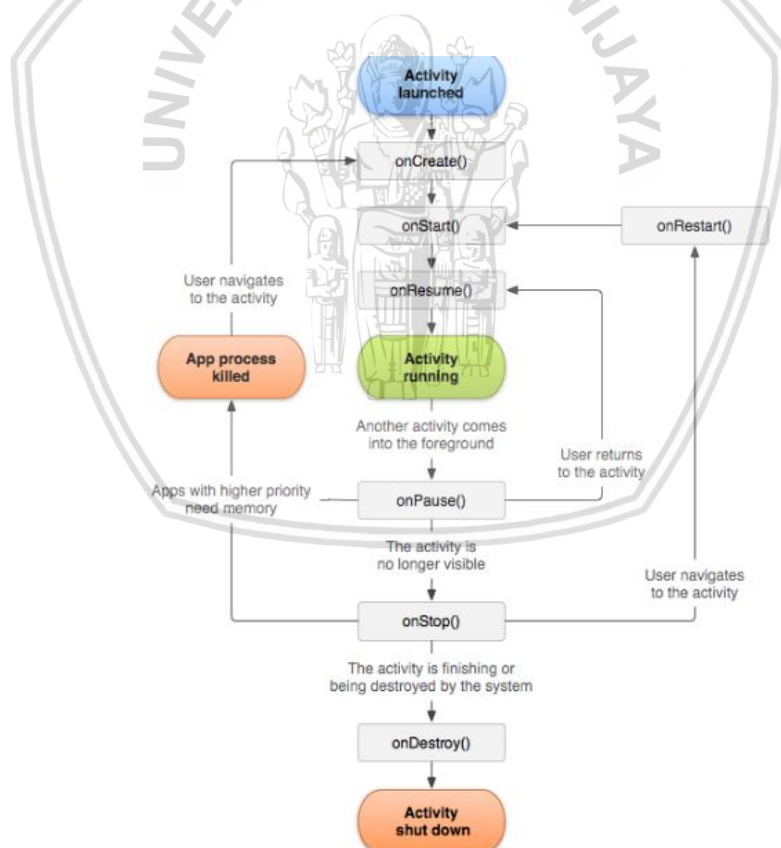
2.5 Android

Android adalah sistem operasi yang dikeluarkan oleh Google khususnya untuk *smartphone* dan tablet (Immanudin & Permana, 2017). Ada beberapa hal yang menjadikan sistem operasi Android layak digunakan oleh pengguna dan dikembangkan oleh developer. Android memanjakan penggunanya dengan beberapa fiturnya yang dinilai sangat baik seperti tampilan antarmuka yang bagus dari segi *user interface* ataupun dari segi *user experience*. Android juga dapat digunakan sebagai alat multimedia seperti pemutar music dan video, serta

menggunakan perangkat keras seperti *accelerometer*, *gyroscope* dan sensor lainnya ke dalam aplikasi (Immanudin & Permana, 2017). Dalam aplikasi android memiliki beberapa komponen yang memiliki fungsi dan peranan masing-masing, diantaranya terdapat *activity*, *intent*, *fragment* dimana ketiga komponen adalah yang sering digunakan dalam membuat aplikasi, serta beberapa komponen lain seperti *threads*, *service*, *receiver* dan lain sebagainya.

2.5.1 Activity

Activity merupakan sebuah komponen di Android yang berfungsi untuk menampilkan *user interface* ke layar handset Android pengguna (Immanudin & Permana, 2017). Salah satu contohnya menampilkan daftar pesan pada aplikasi Gmail. Pada umumnya sebuah aplikasi memiliki beberapa *activity* yang saling terhubung untuk menjalankan tugas yang berbeda-beda. Pada implementasinya, *activity* selalu memiliki satu *layout user interface* dalam bentuk file xml. *Activity* memiliki daur hidup (*lifecycle*) dalam sebuah *stack* pada *virtual sandbox* yang disiapkan oleh *Dalvik Virtual Machine (DVM)* atau *Android Runtime (ART)* yang bersifat *last in first out* (Immanudin & Permana, 2017). Untuk mengetahui daur hidup dari *activity* pada aplikasi android dapat dilihat pada Gambar 2.8 berikut.



Gambar 2.8 Activity lifecycle

(Sumber : <https://developer.android.com/reference/android/app/Activity.html>)

Berdasarkan Gambar 2.8 di atas, *activity* melakukan semua inisialisasi komponen pada *method* *onCreate()*, kemudian menjalankan *method* *onStart()* dilanjutkan *method* *onResume()* dan kemudian masuk ke dalam stack *activity*. Ketika berpindah *activity* maka *activity* pertama akan memanggil *method* *onStop()*. Ketika pengguna kembali ke *activity* pertama maka *rangkaian callback method* yang terpanggil adalah dari *onStop()* -> *onRestart()* -> *onResume()*. Ketika *activity* sudah tidak berjalan dalam jangka waktu tertentu dan sistem menganggap *activity* telah selesai dijalankan maka *activity* menjalankan *method* *onDestroy()*.

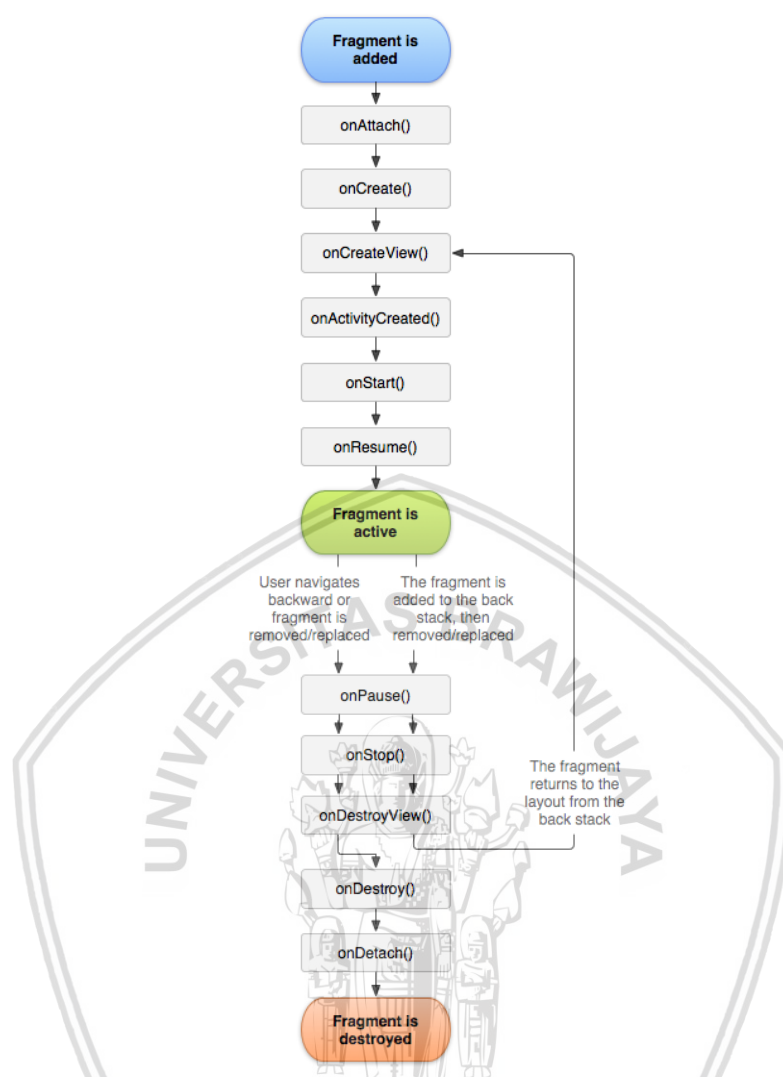
2.5.2 Intent

Intent merupakan mekanisme untuk melakukan sebuah aksi dan komunikasi antar komponen aplikasi pada platform android, termasuk di dalamnya terdapat pengiriman ataupun pengambilan data jika diperlukan (Immanudin & Permana, 2017). *Intent* terbagi dalam dua jenis yaitu :

1. Eksplisit intent, yaitu tipe *intent* yang secara eksplisit mencantumkan nama kelas yang dituju. *Intent* jenis ini berguna untuk mengaktifkan kelas lain sebagai komponen pada aplikasi yang sama.
2. Implicit Intent, yaitu tipe *intent* yang digunakan untuk menjalankan fitur dari komponen aplikasi lain dalam satu *device*, sehingga komponen dari aplikasi lain menjalankan fitur yang yang dipanggil dalam *intent*.

2.5.3 Fragment

Fragment merupakan komponen yang memiliki fungsi untuk menampilkan antar muka pada pengguna melalui *activity* dengan memiliki layout xml tersendiri (Immanudin & Permana, 2017). *Fragment* memiliki daur hidup sendiri, akan tetapi bergantung penuh pada daur hidup *activity* dimana *fragment* tersebut ditanamkan. Alasan digunakannya *fragment* adalah lebih kepada pemecahan komonen tampilan aplikasi untuk menjadi fleksibel dan dapat digunakan kembali (*reusable*) (Immanudin & Permana, 2017). Beberapa *fragment* dapat menempel pada satu *activity*. Implementasi *fragment* dilakukan dengan membuat sebuah kelas Java yang meng-*extends* kelas *Fragment*. Daur hidup *fragment* dapat dilihat pada Gambar 2.9 berikut.



Gambar 2.9 Daur hidup *fragment*

(Sumber :

<https://developer.android.com/guide/components/fragments.html?hl=id>)

Gambar 2.9 diatas menggambarkan daur hidup *fragment* pada saat *activity* berjalan. Untuk membuat kelas *Fragment* hanya perlu mengimplementasikan setidaknya 3 *method* yaitu *onCreate()*, *onCreateView()*, dan *onPause()*. *Method* *onCreate()* dipanggil saat aplikasi membuat *fragment*. Dalam implementasinya, *method* *onCreate()* berisi komponen penting dari *fragment* yang harus dipertahankan saat *fragment* dihentikan sementara dan dilanjutkan kembali. Dalam *method* *onCreateView()*, antarmuka pengguna digambarkan untuk yang pertama kali. Dalam *method* ini harus mengembalikan *View* yang akan menjadi *root layout fragment*. Sistem akan memanggil *method* *onPause()* ketika pengguna meninggalkan *fragment*. *Method* *onPause()* digunakan untuk menyimpan setiap perubahan yang ingin dipertahankan sebelum pengguna meninggalkan *fragment*, sehingga ketika *fragment* dijalankan kembali tidak perlu lagi melakukan inisialisasi data dari awal.

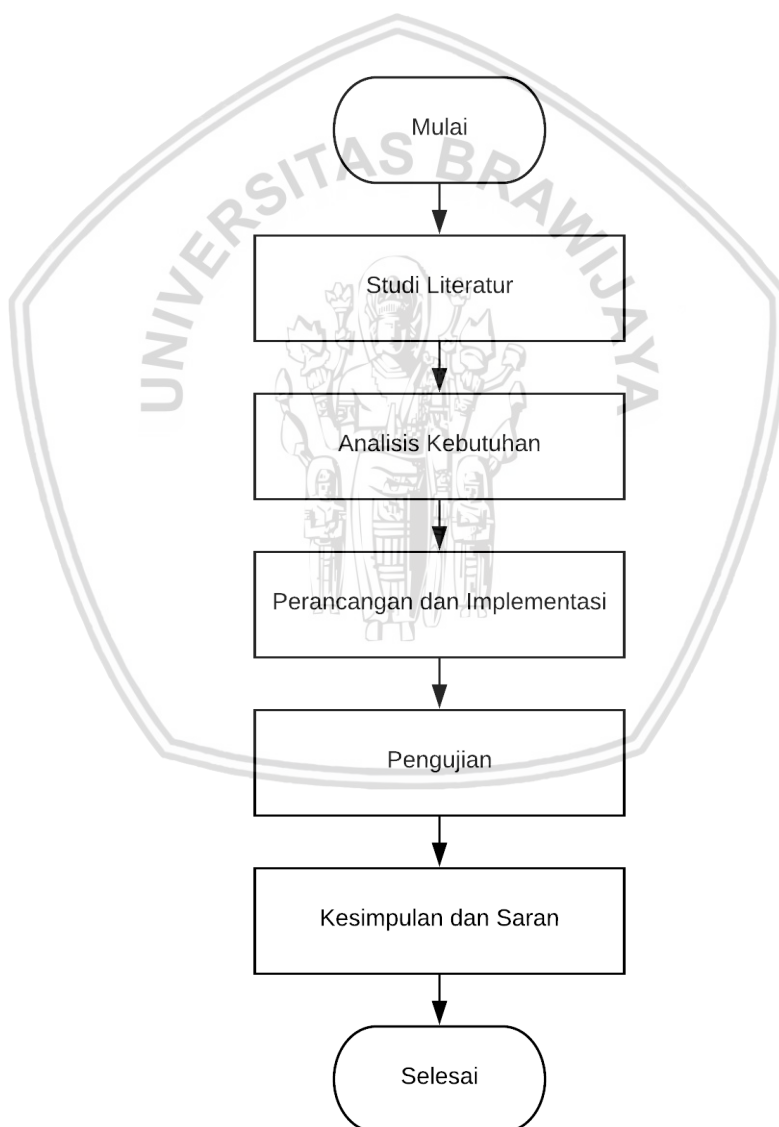
Fragment dapat berupa dalam 3 status diantaranya adalah (*Resumed*, *Paused* dan *Stopped*). Penjelasannya adalah sebagai berikut:

1. *Resumed*, yaitu kondisi dimana *fragment* bias dilihat ketika *activity* sedang berjalan.
2. *Paused*, yaitu kondisi dimana ada *activity* lain yang menutupi sebagian dari *activity* dimana *fragment* ditambahkan. Sebuah *activity* dikatakan tertutup sebagian apabila ada bagian dari *activity* tersebut yang masih dapat dilihat pada layar.
3. *Stopped*, yaitu kondisi ketika *fragment* tidak dapat dilihat pada layar. Kondisi ini bias disebabkan karena *activity* dimana *fragment* itu ditambahkan mengalami *state stopped* atau bahkan *fragment* itu sendiri sudah di hapus dari *activity* dan dilemparkan ke *back stack*. Pada kondisi ini *fragment* masih hidup dengan semua informasinya, tetapi sudah tidak terlihat pada layar dan selanjutnya di *destroy* ketika *activity*-nya di *destroy*



BAB 3 METODOLOGI

Metode penelitian menjelaskan mengenai langkah-langkah yang dilakukan dalam penelitian. Dalam penelitian terkait pengembangan aplikasi pembelajaran pemrograman Java berbasis android ini, beberapa langkah yang dilakukan adalah melakukan kajian terhadap studi literatur, melakukan proses analisis kebutuhan, melakukan perancangan berdasarkan hasil dari analisis kebutuhan yang dilakukan dan mengimplementasikan hasil rancangan yang diperoleh, melakukan pengujian terhadap aplikasi yang telah dibangun dan melakukan penarikan kesimpulan berdasarkan hasil dari seluruh proses pengembangan aplikasi yang dilakukan. Tahapan-tahapan dalam penelitian dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Digram Alir Metodologi Penelitian

3.1 Studi Literatur

Studi literatur dilakukan dengan mengkaji teori, data dan informasi pendukung penelitian dari beberapa sumber. Sumber yang dimaksud dapat berupa buku referensi, jurnal, penelitian sebelumnya yang terkait serta artikel dari sumber yang dapat dipertanggungjawabkan. Teori yang digunakan untuk mendukung proses penelitian yang dilakukan adalah terkait teori pengembangan perangkat lunak, teori terkait *web service* dan teori pengembangan aplikasi android. Data yang diperlukan dalam penelitian ini adalah data dari penelitian yang berjudul "*Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*".

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk memperoleh seluruh kebutuhan yang diperlukan oleh aplikasi. Pada penelitian ini, aplikasi dibangun dengan menggunakan pendekatan berorientasi objek, sehingga metode analisis kebutuhan yang digunakan adalah *OOA (Object Oriented Analysis)*. Proses analisis kebutuhan dilakukan dalam 2 tahap, yaitu:

1. Elisitasi dan spesifikasi kebutuhan.

Pada tahap ini kebutuhan diperoleh dengan cara melakukan analisis pada sistem pembelajaran pemrograman Java berbasis *website* untuk menemukan fitur-fitur utamanya. Proses analisis dilakukan dengan menemukan fitur yang ditujukan kepada pengguna. Fitur-fitur temuan akan dipetakan menjadi daftar temuan kebutuhan fungsional dan non-fungsional dari aplikasi pembelajaran pemrograman Java berbasis android. Selanjutnya, daftar kebutuhan di spesifikasikan untuk menjadi spesifikasi kebutuhan aplikasi. Temuan kebutuhan juga digunakan untuk menentukan kebutuhan *web service*.

2. Pemodelan kebutuhan

Pemodelan kebutuhan dilakukan dengan memodelkan seluruh kebutuhan yang diperoleh ke dalam diagram *use case* dan *use case scenario*. Diagram *use case* menjelaskan hubungan antara aktor dengan sistem, sedangkan *use case scenario* digunakan untuk menjelaskan tiap proses dalam *use case* menjadi lebih rinci.

3.3 Perancangan dan Implementasi

Perancangan aplikasi dilakukan menggunakan metode *OOD (Object Oriented Design)*. Perancangan aplikasi dilakukan berdasarkan hasil dari tahap analisis kebutuhan yang dilakukan sebelumnya. Tahapan perancangan digunakan sebagai dasar dari proses implementasi aplikasi. Tahapan-tahapan dalam melakukan perancangan adalah sebagai berikut:

1. Perancangan Arsitektur

Perancangan arsitektur dilakukan dalam dua tahap, yaitu perancangan *sequence diagram* untuk menggambarkan interaksi antar objek dalam aplikasi beserta pesan yang dikirim atau diterima dan perancangan *class diagram* untuk menggambarkan kelas-kelas yang menyusun aplikasi.

2. Perancangan Komponen

Pada perancangan komponen dilakukan perancangan algoritme dari fungsi-fungsi yang menyusun aplikasi. Dalam laporan penelitian ini akan dibahas 3 algoritme sebagai *sample* yang merupakan algoritme dari fitur utama pada aplikasi dalam bentuk *pseudocode*.

3. Perancangan Antarmuka

Perancangan antarmuka dilakukan dengan menggambarkan susunan komponen dalam antarmuka yang menjadi dasar dalam mengimplementasikan antarmuka aplikasi.

4. Perancangan Web Service

Perancangan *web service* dilakukan dengan melakukan perancangan *class diagram* untuk menggambarkan kelas-kelas yang menyusun *web service*.

Tahapan implementasi dilakukan dengan mengacu pada hasil perancangan yang telah dilakukan. Tahapan-tahapan implementasi adalah sebagai berikut:

1. Implementasi Kode Program

Proses implementasi kode program dilakukan dengan menterjemahkan algoritme hasil perancangan ke dalam kode program. Implementasi dilakukan dalam bahasa pemrograman Java menggunakan android studio.

2. Implementasi Antarmuka Aplikasi

Proses implementasi antarmuka aplikasi dilakukan dengan mengacu pada hasil perancangan antarmuka. Implementasi dilakukan dalam format XML menggunakan android studio.

3. Implementasi Web Service

Proses implementasi *web service* dilakukan dengan mengimplementasikan seluruh *method* pada *class diagram* yang telah digambarkan pada tahap perancangan *web service* ke dalam kode program.

3.4 Pengujian

Pengujian dilakukan untuk memastikan bahwa aplikasi yang dikembangkan sesuai dengan kebutuhan yang telah didefinisikan dan seluruh fitur pada aplikasi telah berjalan sesuai dengan fungsionalitasnya. Tahapan dalam pengujian adalah sebagai berikut:

1. Pengujian Unit

Pengujian unit dilakukan menggunakan metode *white box*. Pengujian *white box* dilakukan untuk mengecek apakah komponen-komponen terkecil pada aplikasi yaitu kelas dan method-method di dalamnya sudah berjalan sebagaimana mestinya.

2. Pengujian Validasi

Pengujian validasi dilakukan menggunakan metode *black box*. Pengujian *black box* dilakukan untuk memastikan bahwa seluruh fungsionalitas dari aplikasi sudah berjalan sesuai kebutuhan yang telah didefinisikan sebelumnya. Pengujian validasi juga dilakukan untuk menguji kebutuhan non fungsional *compatibility*. Pengujian kompatibilitas dilakukan untuk memastikan aplikasi yang dibangun dapat berjalan pada beberapa versi sistem operasi android

3.5 Kesimpulan dan Saran

Penarikan kesimpulan dilakukan berdasarkan hasil dari tahap studi literatur, analisis kebutuhan, perancangan dan implementasi dan pengujian, sehingga kesimpulan diambil setelah proses-proses diatas telah selesai dilakukan. Kesimpulan dilakukan untuk menjawab permasalahan yang telah dirumuskan pada bagian pendahuluan. Saran dituliskan berdasarkan kekurangan dari penelitian yang dilakukan, berupa kekurangan aplikasi ataupun penambahan fitur tertentu untuk pengembangan lebih lanjut.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini menjelaskan mengenai Gambaran umum aplikasi yang akan dibangun, aktor yang akan menggunakan aplikasi serta kebutuhan dalam membangun aplikasi.

4.1 Gambaran Umum Aplikasi

Gambaran umum aplikasi pembelajaran pemrograman Java berbasis android ini dibagi menjadi dua bagian, yaitu deskripsi umum aplikasi dan lingkungan aplikasi.

4.1.1 Deskripsi Umum Aplikasi

Dalam peneitian ini dibangun sebuah aplikasi pembelajaran pemrograman Java yang atraktif berbasis android. Aplikasi ini dibangun sebagai penelitian lanjutan dari penelitian sebelumnya yang dilakukan oleh Bastari, et al. (2017) dengan judul "*Pengembangan Aplikasi Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*". Sistem pembelajaran pemrograman Java tersebut dibangun untuk memudahkan mahasiswa mempelajari materi pemrograman Java secara atraktif dalam permainan simulasi menggunakan metode *gamification*. Tujuannya untuk meningkatkan minat mahasiswa dalam mempelajari pemrograman Java. Sebagai penelitian lanjutan, pada penelitian ini akan dibangun aplikasi pembelajaran pemrograman Java dalam versi aplikasi *mobile* berbasis android. Fitur dari versi aplikasi *mobile* tidak jauh berbeda dengan versi *website*. Semua fitur utama pada *website* akan menjadi fitur-fitur dalam aplikasi *mobile* dan keduanya akan dihubungkan menggunakan *web service* sebagai *Application Programming Interface* (API). API digunakan untuk menjembatani pertukaran data antara aplikasi yang dibangun (*client*) dan sistem berbasis *website* yang sudah ada (*server*).

4.1.2 Lingkungan Sistem

Aplikasi pembelajaran pemrograman Java ini dikembangkan dalam platform android dalam bahasa pemrograman Java menggunakan *Integrated Development Environtment* (IDE) android studio. Sedangkan untuk *web service* ditambahkan pada sistem pembelajaran pemrograman Java berbasis *website* yang dibangun menggunakan bahasa pemrograman PHP menggunakan *framework CodeIgniter* versi 3.1.3.

4.2 Proses Rekayasa Kebutuhan

4.2.1 Elisitasi dan Spesifikasi Kebutuhan

Pada pengembangan aplikasi pembelajaran pemrograman Java berbasis android, proses elisitasi kebutuhan dilakukan dengan melakukan analisis terhadap sistem pembelajaran pemrograman Java berbasis *website* yang sudah ada. Analisis dilakukan dengan memetakan fitur-fitur utama pada *website* untuk

menjadi fitur-fitur utama pada aplikasi berbasis android. Dari hasil analisis yang dilakukan, temuan kebutuhan aplikasi dijabarkan pada Tabel 4.1 berikut.

Tabel 4.1 Temuan Kebutuhan

| No | Deskripsi |
|----|---|
| 1 | Registrasi diperlukan sebagai syarat agar pengguna dapat terdaftar di dalam sistem sebagai member, dapat melakukan <i>login</i> serta <i>logout</i> . |
| 2 | Terdapat halaman profil yang berisi informasi data diri Member, data Member lain dengan ranking teratas, serta daftar riwayat latihan, tantangan dan penghargaan yang diperoleh Member. |
| 3 | Member dapat melihat materi terkait pemrograman dengan kategori tertentu dan dapat melakukan latihan secara langsung pada aplikasi. |
| 4 | Member dapat melihat daftar Member lain yang memiliki hubungan pertemanan serta dapat mengirimkan permintaan pertemanan pada Member lain. |
| 5 | Member dapat menantang Member lain yang sudah tergabung dalam hubungan pertemanan. |

4.2.1.1 Identifikasi Aktor

Aktor adalah seseorang ataupun sistem lain yang mengakses dan berinteraksi dengan sistem. Dalam aplikasi pembelajaran pemrograman Java ini terdapat 2 aktor yang dijelaskan pada Tabel 4.2 berikut

Tabel 4.2 Identifikasi Aktor

| Aktor | Deskripsi |
|----------|--|
| Pengguna | Dalam aplikasi ini pengguna adalah aktor yang belum terautentikasi dalam sistem. Pengguna perlu melakukan registrasi dan <i>login</i> untuk bisa mengakses aplikasi. |
| Member | Member adalah pengguna yang telah berhasil melakukan autentikasi pada sistem. Aktor Member memiliki otoritas untuk dapat menjalankan semua fitur pada aplikasi. |

4.2.1.2 Daftar Kebutuhan Fungsional

Kebutuhan Fungsional adalah kebutuhan yang harus tersedia pada sistem dan merepresentasikan fungsi utama dari sistem. Setiap kebutuhan memiliki kode unik sebagai nomor kebutuhan. Aturan penomoran kebutuhan akan dijelaskan pada Gambar 4.1 dan Tabel 4.3 berikut

CMA-F/NF-XX-YY

Gambar 4.1 Aturan Penomoran Kebutuhan

Tabel 4.3 Penjelasan Aturan Penomoran Kebutuhan

| Aturan Penomoran | Deskripsi |
|------------------|--|
| CMA | Merupakan nama dari aplikasi yang akan dibangun yaitu Code Maniac Application. |
| F/NF | Menyatakan jenis kebutuhan, F untuk kebutuhan fungsional dan NF untuk kebutuhan non-fungsional |
| XX | Menyatakan jenis kebutuhan berdasarkan aktor (hanya untuk kebutuhan fungsional) |
| YY | Menyatakan nomor urut kebutuhan |

4.2.1.3 Kebutuhan Fungsional Pengguna

Tabel 4.4 Daftar Kebutuhan Fungsional Pengguna

| No | Kode Kebutuhan | Deskripsi | Nama Use Case |
|----|----------------|--|---------------|
| 1 | CMA-F-1-001 | Aplikasi dapat menyimpan data registrasi pengguna. | Register |
| | | 1. Aplikasi dapat menampilkan form registrasi pengguna. (CMA-1-001-01) 2. Aplikasi dapat menyimpan data registrasi pengguna pada database. (CMA-1-001-02) | |
| 2 | CMA-F-1-002 | 1. Aplikasi dapat melakukan autentikasi bagi Member. | Login |
| | | 1. Aplikasi dapat menampilkan form login Member. (CMA-1-002-01) 2. Aplikasi dapat melakukan validasi username dan password Member. (CMA-1-002-02) | |

4.2.1.4 Kebutuhan Fungsional Member

Tabel 4.5 Daftar Kebutuhan Fungsional Member

| No | Kode Kebutuhan | Deskripsi | Nama Use Case |
|----|----------------|--|----------------|
| 1 | CMA-F-2-003 | Aplikasi dapat menampilkan informasi profil. | Melihat profil |

| No | Kode Kebutuhan | Deskripsi | Nama Use Case |
|----|----------------|---|----------------------------|
| | | 1. Aplikasi dapat menampilkan informasi profil Member. (CMA-1-003-01) 2. Aplikasi dapat menampilkan daftar latihan Member. (CMA-1-003-02) 3. Aplikasi dapat menampilkan daftar tantangan Member. (CMA-1-003-03) 4. Aplikasi dapat menampilkan daftar penghargaan Member. (CMA-1-003-04) | |
| 2 | CMA-F-2-004 | Aplikasi dapat menyimpan perubahan profil. 1. Aplikasi dapat menampilkan form perubahan profil Member. (CMA-1-004-01) 2. Aplikasi dapat melakukan perubahan foto profil Member. (CMA-1-004-02) 3. Aplikasi dapat menyimpan data perubahan profil pada <i>database</i> . (CMA-1-004-03) | Melakukan Perubahan Profil |
| 3 | CMA-F-2-005 | Aplikasi dapat menampilkan informasi detail latihan. 1. Aplikasi dapat menampilkan detail informasi latihan yang dikerjakan Member. (CMA-1-005-01) | Melihat Detail Latihan |
| 4 | CMA-F-2-006 | Aplikasi dapat menampilkan informasi detail tantangan. 1. Aplikasi dapat menampilkan daftar tantangan yang diperoleh Member. (CMA-1-006-01) 2. Aplikasi dapat menampilkan daftar tantangan yang diajukan Member kepada Member lain. (CMA-1-006-02) 3. Aplikasi dapat menampilkan daftar tantangan yang telah dikerjakan oleh Member beserta hasilnya. (CMA-1-006-03) | Melihat Detail Tantangan |
| 5 | CMA-F-2-007 | Aplikasi dapat menampilkan daftar Member lain yang memiliki hubungan pertemanan. 1. Aplikasi dapat menampilkan daftar Member lain yang memiliki hubungan pertemanan. (CMA-1-007-01) | Melihat Daftar Teman |

| No | Kode Kebutuhan | Deskripsi | Nama Use Case |
|----|----------------|--|--------------------------------------|
| 6 | CMA-F-2-008 | Aplikasi dapat mengirimkan permintaan pertemanan kepada Member lain. | Mengirim Permintaan Pertemanan |
| | | 1. Aplikasi dapat menampilkan daftar Member lain yang tidak memiliki hubungan pertemanan dengan Member. (CMA-1-008-01) | |
| | | 2. Aplikasi dapat mengirimkan permintaan pertemanan kepada Member lain. (CMA-1-008-02) | |
| 7 | CMA-F-2-009 | Aplikasi dapat menampilkan daftar permintaan pertemanan. | Melihat Daftar Permintaan Pertemanan |
| | | 1. Aplikasi dapat menampilkan daftar permintaan pertemanan dari Member lain. (CMA-1-009-01) | |
| 8 | CMA-F-2-010 | Aplikasi dapat menyediakan layanan untuk menerima permintaan pertemanan. | Menerima Permintaan Pertemanan |
| | | 1. Aplikasi dapat menyediakan layanan untuk menerima permintaan pertemanan dari Member lain. (CMA-1-010-01) | |
| 9 | CMA-F-2-011 | Aplikasi dapat menyediakan fungsi pengajuan tantangan kepada Member lain yang memiliki hubungan pertemanan. | Mengajukan Tantangan |
| | | 1. Aplikasi dapat menyediakan pilihan kategori tantangan. (CMA-1-011-01) | |
| | | 2. Aplikasi dapat menyediakan pilihan soal tantangan. (CMA-1-011-02) | |
| | | 3. Aplikasi dapat melanjutkan pengajuan tantangan. (CMA-1-011-03) | |
| 10 | CMA-F-2-012 | Aplikasi dapat menampilkan materi latihan. | Melihat Materi Latihan |
| | | 1. Aplikasi dapat menyediakan pilihan kategori latihan. (CMA-1-012-01) | |
| | | 2. Aplikasi dapat menyediakan pilihan daftar latihan. (CMA-1-012-01) | |
| | | 3. Aplikasi dapat menampilkan informasi materi latihan. (CMA-1-012-03) | |
| 11 | CMA-F-2-013 | Aplikasi dapat menyediakan layanan mengerjakan soal latihan. | Mengerjakan Latihan |
| | | 1. Aplikasi dapat menyediakan pilihan soal latihan. (CMA-1-013-01) | |
| | | 2. Aplikasi dapat menyediakan halaman | |

| No | Kode Kebutuhan | Deskripsi | Nama Use Case |
|----|----------------|--|-------------------------------|
| | | pengerjaan latihan. (CMA-1-013-02) 3. Aplikasi dapat menyimpan data jawaban latihan pada <i>database</i> . (CMA-1-013-03) | |
| 12 | CMA-F-2-014 | Aplikasi dapat menyediakan layanan mengerjakan soal tantangan. 1. Aplikasi dapat menyediakan halaman pengerjaan tantangan. (CMA-1-014-01) 2. Aplikasi dapat menyimpan data jawaban tantangan pada <i>database</i> . (CMA-1-014-02) | Mengerjakan Tantangan |
| 13 | CMA-F-2-015 | Aplikasi dapat melakukan kompilasi kode Java. 1. Aplikasi dapat melakukan kompilasi kode Java. (CMA-1-015-01) 2. Aplikasi dapat menampilkan hasil kompilasi. (CMA-1-015-02) | Melakukan Kompilasi Kode Java |
| 14 | CMA-F-2-016 | Aplikasi dapat melakukan percobaan kode Java. 1. Aplikasi dapat melakukan percobaan kode Java. (CMA-1-016-01) 2. Aplikasi dapat menampilkan hasil percobaan kode Java. (CMA-1-016-02) | Melakukan Percobaan Kode Java |
| 15 | CMA-F-2-017 | Aplikasi dapat menyediakan layanan menjawab tantangan. 1. Aplikasi dapat menyediakan layanan menjawab soal tantangan yang diperoleh Member. (CMA-1-017-01) | Menjawab Tantangan |
| 16 | CMA-F-2-018 | Aplikasi dapat menyediakan layanan untuk menolak permintaan pertemanan. 1. Aplikasi dapat menyediakan layanan untuk menolak permintaan pertemanan dari Member lain. (CMA-1-018-01) | Menolak Permintaan Pertemanan |
| 17 | CMA-F-2-019 | Aplikasi dapat menyediakan layanan bagi Member keluar dari akun aplikasi. | Logout |

4.2.1.5 Daftar Kebutuhan Non Fungsional

Tabel 4.6 Daftar Kebutuhan Non Fungsional

| No | Kode Kebutuhan | Deskripsi | Nama Kebutuhan |
|----|----------------|---|----------------------|
| 1 | CMA-NF-001 | Aplikasi dapat berjalan pada beberapa versi sistem operasi android, yaitu dari android versi 5 (kitkat) sampai dengan android versi 8 (oreo) dan pada beberapa ukuran layar, yaitu dari rentan 4,7 inchi sampai dengan 6 inchi. | <i>Compatibility</i> |

4.2.1.6 Daftar Kebutuhan Web Service

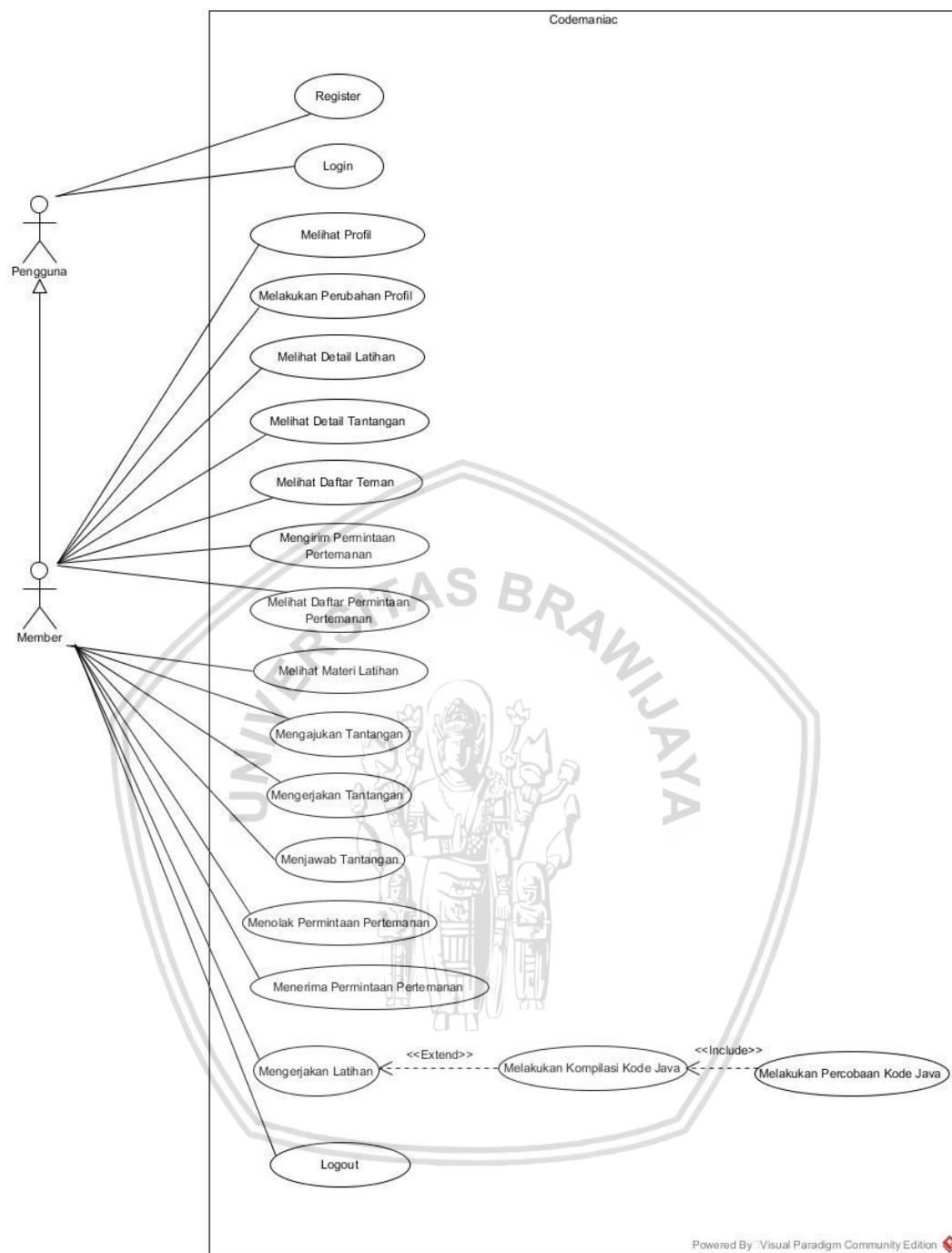
| No | Deskripsi | Nama Kebutuhan |
|----|--|---|
| 1 | Sistem dapat menyimpan data registrasi pengguna | <i>Register</i> |
| 2 | Sistem dapat melakukan autentikasi bagi member | <i>Login</i> |
| 3 | Sistem dapat menyediakan data profil | Menyediakan data profil |
| 4 | Sistem dapat menyimpan data perubahan profil | Menyimpan Perubahan Profil |
| 5 | Sistem dapat menyediakan data detail latihan | Menyediakan Data Detail Latihan |
| 6 | Sistem dapat menyediakan data detail tantangan | Menyediakan Data Detail Tantangan |
| 7 | Sistem dapat menyediakan data daftar Member lain yang memiliki hubungan pertemanan. | Menyediakan Data Daftar Teman |
| 8 | Sistem dapat mengirimkan permintaan pertemanan kepada member lain | Mengirim Permintaan Pertemanan |
| 9 | Sistem dapat menyediakan data daftar permintaan pertemanan | Menyediakan Data Daftar Permintaan Pertemanan |
| 10 | Sistem dapat menyediakan layanan untuk menerima permintaan pertemanan | Menerima Permintaan Pertemanan |
| 11 | Sistem dapat menyediakan fungsi pengajuan tantangan kepada Member lain yang memiliki hubungan pertemanan | Mengajukan Tantangan |
| 12 | Sistem dapat menyediakan data materi latihan | Menyediakan Data Materi Latihan |
| 13 | Sistem dapat menyimpan data jawaban latihan | Menyimpan Jawaban Latihan |
| 14 | Sistem dapat menyimpan data jawaban tantangan | Menyediakan Jawaban Tantangan |

| No | Deskripsi | Nama Kebutuhan |
|----|--|-------------------------------|
| 15 | Sistem dapat melakukan kompilasi kode Java | Melakukan Kompilasi Kode Java |
| 16 | Sistem dapat melakukan percobaan kode Java | Melakukan Percobaan Kode Java |
| 17 | Sistem dapat menyediakan layanan menjawab tantangan | Menjawab Tantangan |
| 18 | Sistem dapat menyediakan layanan untuk menolak permintaan pertemanan | Menolak Permintaan Pertemanan |

4.2.2 Pemodelan Diagram *Use case*

Diagram *use case* digunakan untuk memodelkan interaksi antara aktor dengan sistem. Pemodelan didasarkan pada daftar kebutuhan fungsional yang diperoleh dan digambarkan dalam bentuk interaksi antara aktor dengan fungsi sistem ataupun antara sistem dengan sistem lain. Pemodelan diagram *use case* dapat dilihat pada Gambar 4.2 berikut.





Gambar 4.2 Diagram Use case

Gambar 4.2 diatas merupakan diagram *use case* dari aplikasi pembelajaran pemrograman Java. Pada aplikasi memiliki 2 aktor yaitu Pengguna dan Member. Pengguna bisa melakukan *Register* dan *Login*. Proses *Register* dilakukan oleh pengguna untuk melakukan registrasi pada sistem. Proses *Login* dilakukan oleh pengguna agar dapat terautentikasi pada aplikasi. *Use case* *Melihat Profil* dilakukan Member untuk melihat informasi profil. *Use case* *Melakukan Perubahan Profil* memungkinkan Member mengubah informasi profil jika diinginkan terjadi perubahan. Pada *use case* *Melihat Detail Latihan* Member

dapat melihat detail informasi masing-masing latihan yang pernah dikerjakan. Pada *use case* Melihat Detail Tantangan Member dapat melihat detail informasi dari masing-masing tantangan yang diperoleh ataupun diajukan pada Member lain. Pada *use case* Melihat Daftar Teman Member dapat melihat daftar dari Member lain yang memiliki hubungan pertemanan dengan Member. Pada *use case* Mengirimkan Permintaan Pertemanan Member dapat mengirimkan permintaan pertemanan kepada Member lain. Pada *use case* Melihat Daftar Permintaan Pertemanan Member dapat melihat daftar Member lain yang mengajukan permintaan pertemanan kepada Member. Pada *use case* Melihat Materi Latihan Member dapat melihat informasi materi latihan sebagai penunjang saat mengerjakan soal latihan. Pada *use case* Mengajukan Tantangan Member dapat mengajukan tantangan pada Member lain untuk mengerjakan soal tertentu yang tersedia dalam aplikasi. Pada *use case* Mengerjakan Tantangan Member dapat mengerjakan soal tantangan dan melakukan *submit*. Pada *use case* Menjawab Tantangan Member dapat menerima tantangan dari member lain dan mengerjakan soal tantangan yang diajukan. Pada *use case* Menerima Permintaan Pertemanan Member dapat melakukan persetujuan permintaan pertemanan dari Member lain. Pada *use case* Menolak Permintaan Pertemanan Member dapat menolak permintaan pertemanan dari Member lain. Pada *use case* Mengerjakan latihan Member dapat mengerjakan soal latihan dan melakukan *submit*. Pada *use case* Melakukan Kompilasi Kode Java Member dapat melakukan kompilasi terhadap kode program Java yang dikerjakan untuk menjawab soal latihan. *Use case* Melakukan Kompilasi Kode Java merupakan *extends* dari *use case* Mengerjakan Latihan, yang berarti *use case* Melakukan Kompilasi Kode Java hanya dapat dilakukan saat Member Mengerjakan Latihan. Pada *use case* Melakukan Percobaan Kode Java Member dapat melakukan percobaan terhadap kode program Java yang dikerjakan untuk menjawab soal latihan untuk mengetahui keluaran kode program. *Use case* Melakukan Percobaan Kode Java merupakan *extends* dari *use case* Melakukan Kompilasi Kode Java, dimana percobaan kode Java hanya dapat dilakukan saat proses kompilasi kode Java berhasil dilakukan.

4.2.3 Skenario *Use case*

Skenario *use case* merupakan deskripsi secara mendetail dari *use case* diagram. Dalam *use case scenarion* berisi nomor *use case*, nama *use case*, nama aktor, tujuan, *pre-condition*, *main flow*, *alternative flow* dan *post condition*. Bagian *pre-condition* berisi kondisi yang harus dicapai sebelum menjalankan *main flow*. Bagian *main flow* mendefinisikan alur proses secara berurutan. Bagian *alternative flow* berisi alur kondisi tertentu yang tidak dideklarasikan pada *main flow*. Bagian terakhir yaitu *post-condition* berisi kondisi akhir setelah aktor menjalankan proses pada *main flow*. Skenario *use case* aplikasi pembelajaran pemrograman Java ditunjukkan pada Tabel 4.6 sampai dengan Tabel 4.21 berikut.

Tabel 4.7 Skenario Use case Register

| | |
|--------------------|---|
| Nomor Use case | CMA-F-1-001 |
| Nama | <i>Register</i> |
| Aktor | Pengguna |
| Tujuan | Menyediakan layanan bagi pengguna melakukan registrasi pada aplikasi |
| Pre-condition | Halaman registrasi telah tersedia |
| Main Flow | <ol style="list-style-type: none"> 1. Pengguna mengisi <i>username</i>, <i>fullname</i>, <i>email</i>, <i>organization</i> dan <i>password</i> pada form registrasi 2. Pengguna menekan tombol registrasi 3. Aplikasi melakukan validasi data registrasi 4. Aplikasi menyimpan data registrasi pengguna pada database |
| Alternative Flow 1 | <ol style="list-style-type: none"> 1. Aktor memasukkan <i>username</i> yang sudah terdaftar 2. Pengguna menekan tombol registrasi 3. Aplikasi gagal menyimpan data registrasi dan menampilkan pesan “username telah terdaftar” |
| Alternative Flow 2 | <ol style="list-style-type: none"> 1. Aktor tidak mengisi seluruh <i>field</i> pada form registrasi 2. Pengguna menekan tombol registrasi 3. Aplikasi gagal menyimpan data registasi dan menampilkan pesan “silahkan mengisi seluruh <i>field</i> registasi” |
| Post-condition | Data registasi pengguna berhasil tersimpan pada database |

Tabel 4.8 Skenario Use case Login

| | |
|--------------------|--|
| Nomor Use case | CMA-F-1-002 |
| Nama | <i>Login</i> |
| Aktor | <i>Pengguna</i> |
| Tujuan | Menyediakan layanan autentikasi bagi Member |
| Pre-condition | Halaman login telah tersedia |
| Main Flow | <ol style="list-style-type: none"> 1. Pengguna memasukkan <i>username</i> dan <i>password</i> pada <i>form</i> login 2. Aktor menekan tombol <i>login</i> 3. Aplikasi melakukan validasi <i>username</i> dan <i>password</i> pengguna |
| Alternative Flow 1 | <ol style="list-style-type: none"> 1. Pengguna memasukkan <i>Username</i> dan <i>password</i> yang tidak valid |

| | |
|---------------------------|---|
| | 2. Aplikasi gagal melakukan autentikasi dan menampilkan pesan " <i>username</i> dan <i>password</i> tidak valid" |
| <i>Alternative Flow 2</i> | 1. Pengguna tidak mengisi seluruh <i>field</i> pada <i>form login</i> 2. Aplikasi gagal melakukan autentikasi dan menampilkan pesan "silahkan mengisi seluruh <i>field login</i> " |
| <i>Post-condition</i> | Proses autentikasi sebagai Member berhasil, pengguna diarahkan halaman profil aplikasi |

Tabel 4.9 Skenario *Use case* Melihat Profil

| | |
|-------------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-003 |
| Nama | Melihat Profil |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk melihat informasi profil |
| <i>Pre-condition</i> | Proses autentikasi Member berhasil dilakukan |
| <i>Main Flow</i> | 1. Aplikasi menampilkan seluruh informasi profil Member |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Informasi profil berhasil ditampilkan pada halaman profil aplikasi |

Tabel 4.10 Skenario *Use case* Melakukan Perubahan Profil

| | |
|-------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-004 |
| Nama | Melakukan Perubahan Profil |
| Aktor | Member |
| Tujuan | Menyediakan layanan perubahan data profil Member |
| <i>Pre-condition</i> | Halaman profil telah tersedia |
| <i>Main Flow</i> | 1. Member memilih menu edit profil 2. Aplikasi menampilkan halaman edit profil 3. Member mengisi <i>field</i> data profil tertentu yang ingin dirubah 4. Aplikasi menyimpan perubahan data profil Member pada database |
| <i>Alternative Flow</i> | Tidak |
| <i>Post-condition</i> | Data profil terbaru Member berhasil tersimpan pada database |

Tabel 4.11 Skenario *Use case* Melihat Detail Latihan

| | |
|-------------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-005 |
| Nama | Melihat Detail Latihan |
| Aktor | Member |
| Tujuan | Menampilkan informasi detail latihan |
| <i>Pre-condition</i> | Informasi daftar latihan telah tersedia pada halaman profil |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol lihat detail pada salah satu daftar latihan 2. Aplikasi menampilkan informasi detail latihan |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Informasi detail latihan berhasil ditampilkan pada halaman detail latihan |

Tabel 4.12 Skenario *Use case* Melihat Detail Tantangan

| | |
|-------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-006 |
| Nama | Melihat Detail Tantangan |
| Aktor | Member |
| Tujuan | Menampilkan informasi detail tantangan |
| <i>Pre-condition</i> | Informasi daftar tantangan telah tersedia pada halaman profil |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol lihat detail tantangan 2. Aplikasi menampilkan daftar tantangan beserta detail informasi berupa status tantangan dan <i>point</i> yang didapatkan |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Informasi detail tantangan berhasil ditampilkan pada halaman detail tantangan |

Tabel 4.13 Skenario *Use case* Melihat Daftar Teman

| | |
|-----------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-007 |
| Nama | Melihat Daftar Teman |
| Aktor | Member |
| Tujuan | Menampilkan daftar Member lain yang memiliki hubungan pertemanan |
| <i>Pre-condition</i> | Proses autentikasi berhasil dilakukan |

| | |
|-------------------------|---|
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member memilih menu pertemanan 2. Aplikasi menampilkan informasi daftar teman |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Informasi daftar teman berhasil ditampilkan pada halaman pertemanan |

Tabel 4.14 Skenario *Use case* Mengirim Permintaan Pertemanan

| | |
|---------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-008 |
| Nama | Mengirim Permintaan Pertemanan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk mengirimkan permintaan pertemanan kepada Member lain |
| <i>Pre-condition</i> | Halaman pertemanan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member memilih menu cari teman baru 2. Aplikasi menampilkan daftar Member lain yang tidak memiliki hubungan pertemanan 3. Member menekan tombol “tambahkan teman” pada salah satu daftar Member lain yang ingin ditambahkan menjadi teman. 4. Aplikasi menampilkan dialog konfirmasi melanjutkan permintaan pertemanan 5. Member menekan tombol “YES” 6. Aplikasi menyimpan data permintaan pertemanan pada <i>database</i> |
| <i>Alternative Flow 1</i> | <ol style="list-style-type: none"> 1. Member menekan tombol “NO” 2. Aplikasi menutup kotak dialog konfirmasi |
| <i>Post Condition</i> | Data permintaan pertemanan berhasil tersimpan pada <i>database</i> |

Tabel 4.15 Skenario *Use case* Melihat Daftar Permintaan Pertemanan

| | |
|-----------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-009 |
| Nama | Melihat Daftar Permintaan Pertemanan |
| Aktor | Member |
| Tujuan | Menampilkan daftar permintaan pertemanan dari Member lain |
| <i>Pre-condition</i> | Halaman pertemanan telah tersedia |

| | |
|-------------------------|---|
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member memilih menu “lihat permintaan pertemanan” 2. Aplikasi menampilkan daftar permintaan pertemanan dari Member lain |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Daftar permintaan pertemanan dari Member lain berhasil ditampilkan pada halaman permintaan pertemanan |

Tabel 4.16 Skenario *Use case* Menerima Permintaan Pertemanan

| | |
|-------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-010 |
| Nama | Menerima Permintaan Petemanan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk menerima permintaan pertemanan dari Member lain |
| <i>Pre-condition</i> | Halaman permintaan pertemanan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol “accept” 2. Aplikasi menyimpan data pertemanan pada <i>database</i> |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Data pertemanan berhasil tersimpan pada <i>database</i> |

Tabel 4.17 Skenario *Use case* Mengajukan Tantangan

| | |
|-----------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-011 |
| Nama | Mengajukan Tantangan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk mengajukan tantangan kepada member lain dalam daftar teman |
| <i>Pre-condition</i> | Halaman petemanan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol <i>challenge</i> pada daftar teman 2. Aplikasi menampilkan halaman kategori tantangan 3. Member memilih kategori tantangan 4. Aplikasi menampilkan halaman soal tantangan 5. Member memilih soal tantangan 6. Aplikasi menampilkan dialog konfirmasi melanjutkan tantangan 7. Member menekan tombol “YES” 8. Aplikasi menyimpan data tantangan pada <i>database</i> |

| | |
|---------------------------|--|
| <i>Alternative Flow 1</i> | 1. Member menekan tombol “NO” 2. Aplikasi menutup dialog konfirmasi |
| <i>Post-condition</i> | Data tantangan berhasil tersimpan pada <i>database</i> |

Tabel 4.18 Skenario *Use case* Melihat Materi Latihan

| | |
|-------------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-012 |
| Nama | Melihat Materi Latihan |
| Aktor | Member |
| Tujuan | Menampilkan informasi materi latihan |
| <i>Pre-condition</i> | Proses autentikasi Member berhasil dilakukan |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member memilih menu latihan 2. Aplikasi menampilkan halaman kategori latihan 3. Member memilih kategori latihan 4. Aplikasi menampilkan halaman daftar latihan 5. Member memilih daftar latihan 6. Aplikasi menampilkan informasi materi latihan |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Informasi materi latihan berhasil ditampilkan pada halaman materi latihan |

Tabel 4.19 Skenario *Use case* Mengerjakan Latihan

| | |
|-----------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-013 |
| Nama | Mengerjakan Latihan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk menjawab soal latihan |
| <i>Pre-condition</i> | Halaman materi latihan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol “mulai latihan” 2. Aplikasi menampilkan halaman daftar soal latihan 3. Member memilih soal latihan 4. Aplikasi menampilkan halaman untuk menjawab soal latihan 5. Member menuliskan kode program sesuai perintah soal latihan 6. Aktor menekan tombol “submit” |

| | |
|-------------------------|--|
| | 7. Aplikasi menyimpan data latihan pada <i>database</i> |
| <i>Alternative Flow</i> | Perluasan ke <i>use case</i> Melakukan Kompilasi Kode Java |
| <i>Post-condition</i> | Data latihan berhasil tersimpan pada <i>database</i> |

Tabel 4.20 Skenario *Use case* Mengerjakan Tantangan

| | |
|-------------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-014 |
| Nama | Mengerjakan Tantangan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk mengerjakan tantangan |
| <i>Pre-condition</i> | Halaman untuk menjawab soal tantangan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menuliskan kode program sesuai perintah soal tantangan 2. Member menekan tombol "<i>submit</i>" 3. Aplikasi menyimpan data pengerjaan tantangan pada <i>database</i> |
| <i>Alternative Flow</i> | Tidak Ada |
| <i>Post-condition</i> | Data pengerjaan tantangan berhasil tersimpan pada <i>database</i> |

Tabel 4.21 Skenario *Use case* Melakukan Kompilasi Kode Java

| | |
|---------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-015 |
| Nama | Melakukan Kompilasi Kode Java |
| Aktor | Member |
| Tujuan | Melakukan Kompilasi kode program Java |
| <i>Pre-condition</i> | Kode program untuk menjawab soal latihan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol "<i>compile</i>" 2. Aplikasi berhasil melakukan kompilasi kode Java 3. Aplikasi mengaktifkan tombol "<i>test code</i>" |
| <i>Alternative Flow 1</i> | <ol style="list-style-type: none"> 1. Aplikasi gagal melakukan kompilasi 2. Aplikasi menampilkan pesan bahwa terdapat kesalahan kode program |
| <i>Post-condition</i> | Kompilasi kode Java berhasil dilakukan |

Tabel 4.22 Skenario *Use case* Melakukan Percobaan Kode Java

| | |
|-------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-016 |
| Nama | Mengirim Data Percobaan Kode Java |
| Aktor | Member |
| Tujuan | Melakukan Percobaan kode Java |
| <i>Pre-condition</i> | Tombol <i>test code</i> telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol <i>test code</i> 2. Aplikasi Melakukan percobaan kode Java 3. Aplikasi menampilkan hasil keluaran kode program |
| <i>Alternative Flow</i> | Tidak Ada |
| <i>Post-condition</i> | Hasil keluaran kode program berhasil ditampilkan |

Tabel 4.23 Skenario *Use case* Menjawab Tantangan

| | |
|-------------------------|---|
| Nomor <i>Use case</i> | CMA-F-2-017 |
| Nama | Menjawab Tantangan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk menjawab soal tantangan |
| <i>Pre-condition</i> | Halaman detail tantangan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol "<i>answer now</i>" 2. Aplikasi menampilkan halaman menjawab soal tantangan |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Halaman menjawab soal tantangan berhasil ditampilkan |

Tabel 4.24 Skenario *Use case* Menolak Permintaan Petemanan

| | |
|-----------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-018 |
| Nama | Menolak Permintaan Petemanan |
| Aktor | Member |
| Tujuan | Menyediakan layanan untuk menolak permintaan pertemanan dari Member lain |
| <i>Pre-condition</i> | Halaman permintaan pertemanan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Member menekan tombol "<i>decline</i>" 2. Aplikasi menghapus data permintaan pertemanan pada |

| | |
|-------------------------|--|
| | <i>database</i> |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Data permintaan pertemanan pada <i>database</i> berhasil dihapus |

Tabel 4.25 Skenario *Use case Logout*

| | |
|-------------------------|--|
| Nomor <i>Use case</i> | CMA-F-2-019 |
| Nama | Logout |
| Aktor | Member |
| Tujuan | Menyediakan layanan bagi Member untuk keluar dari akun aplikasi |
| <i>Pre-condition</i> | Halaman pengaturan telah tersedia |
| <i>Main Flow</i> | <ol style="list-style-type: none"> 1. Aktor menekan tombol "<i>logout</i>" 2. Aplikasi menghapus data <i>session</i> pada aplikasi |
| <i>Alternative Flow</i> | Tidak ada |
| <i>Post-condition</i> | Data <i>session</i> pada aplikasi berhasil dihapus |

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini dijelaskan mengenai 2 tahap yaitu perancangan dan implementasi. Perancangan sistem merupakan dasar dalam mengimplementasikan sistem. Perancangan sistem dilakukan berdasarkan analisis kebutuhan dan pemodelan kebutuhan yang dilakukan. Sedangkan implementasi dilakukan dengan menerapkan hasil rancangan sistem untuk membangun sistem secara keseluruhan yang terbagi dalam implementasi kode program dan implementasi antarmuka sistem.

5.1 Perancangan

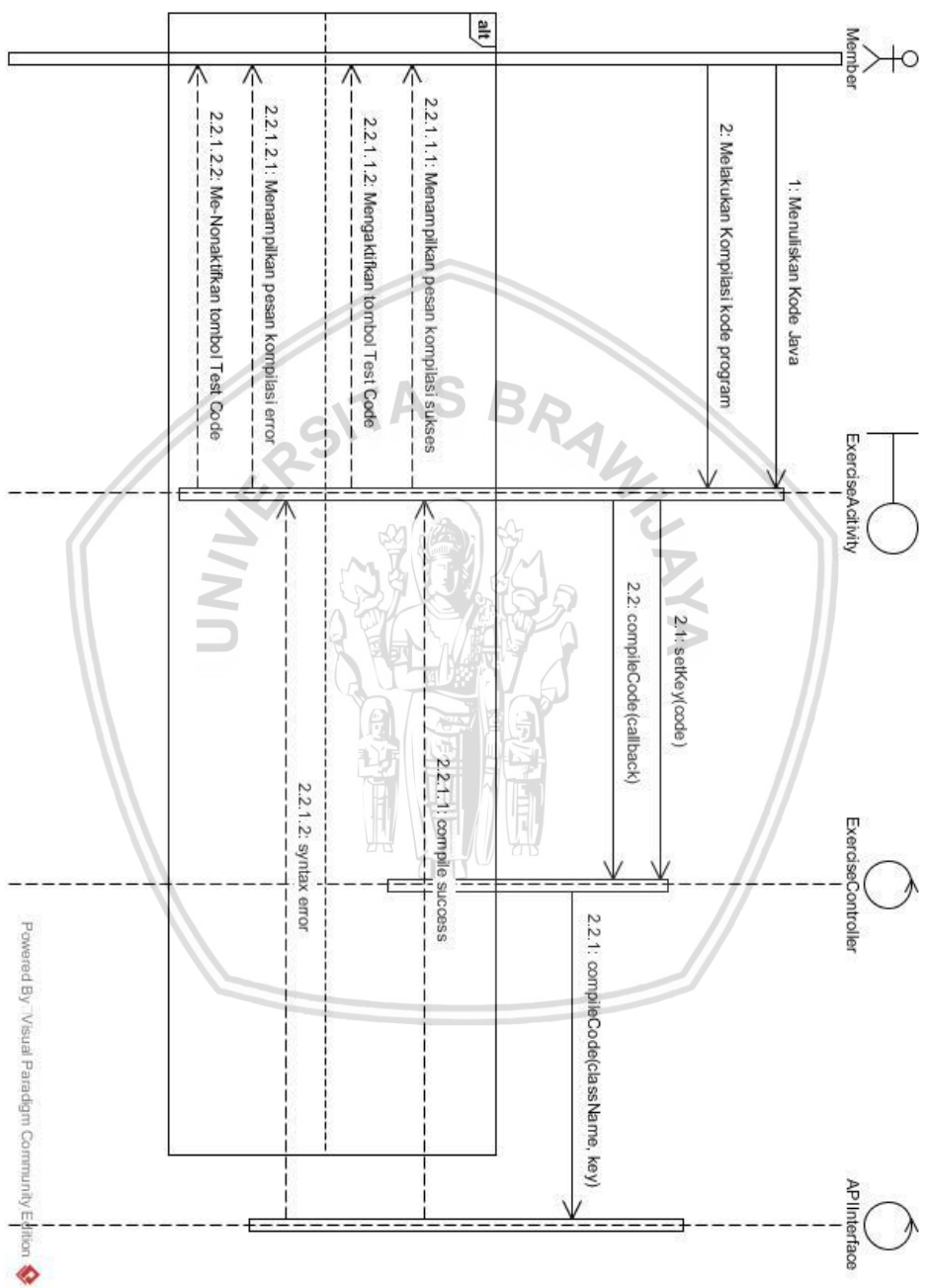
Perancangan dilakukan setelah tahap analisis kebutuhan selesai dilakukan. Hasil dari proses analisis kebutuhan berupa kebutuhan fungsional dan hasil dari pemodelan kebutuhan berupa *use case* dan *use case scenario* dijadikan dasar dalam merancang sistem. Proses perancangan aplikasi pembelajaran pemrograman java ini dibagi ke dalam 3 tahap yakni perancangan arsitektur, perancangan komponen dan perancangan antarmuka.

5.1.1 Perancangan Sequence Diagram

Sequence diagram dibuat dengan tujuan untuk mengetahui alur sistem yang dibagi berdasarkan fungsionalitas serta objek apa saja yang terlibat di dalamnya. Dasar dari pembuatan *sequence diagram* adalah hasil dari skenario *use case* pada masing-masing kebutuhan fungsional yang telah di definisikan pada saat tahap analisis kebutuhan sistem.

5.1.1.1 Sequence Diagram Melakukan Kompilasi Kode Java

Pada Gambar 5.1 berikut, proses melakukan kompilasi kode Java dimulai pada saat Member menekan tombol *compile* setelah selesai menuliskan kode Java pada kelas *TakeExerciseActivity*. Ketika aktor menekan tombol *compile* maka *TakeExerciseActivity* memanggil method *setKey()* dengan kode Java sebagai parameternya pada kelas *ExerciseController*. Selanjutnya *TakeExerciseActivity* memanggil method *compileCode()* pada kelas *ExerciseController* dengan parameter berupa *callback*. *Callback* merupakan sebuah *interface* bawaan dari library *retrofit* yang berisi method *onResponse* untuk menangkap pesan sukses dan *onFailure* untuk menangkap pesan kegagalan saat memanggil API. Kemudian *ExerciseController* memanggil method *compileCode()* pada kelas *APIInterface* dengan parameter nama kelas berserta kode Java yang akan dikompilasi. Pemanggilan API menghasilkan *response* yang berisi informasi hasil kompilasi. Jika kompilasi sukses maka *TakeExerciseActivity* menampilkan pesan sukses dan mengaktifkan tombol *test code* yang digunakan untuk melakukan percobaan kode Java. Jika kompilasi error maka *TakeExerciseActivity* menampilkan pesan kegagalan dan menonaktifkan tombol *test code*.



Gambar 5.1 *Sequence Diagram* Melakukan Kompilasi Kode Java

5.1.1.2 Sequence Diagram Mengajukan Tantangan

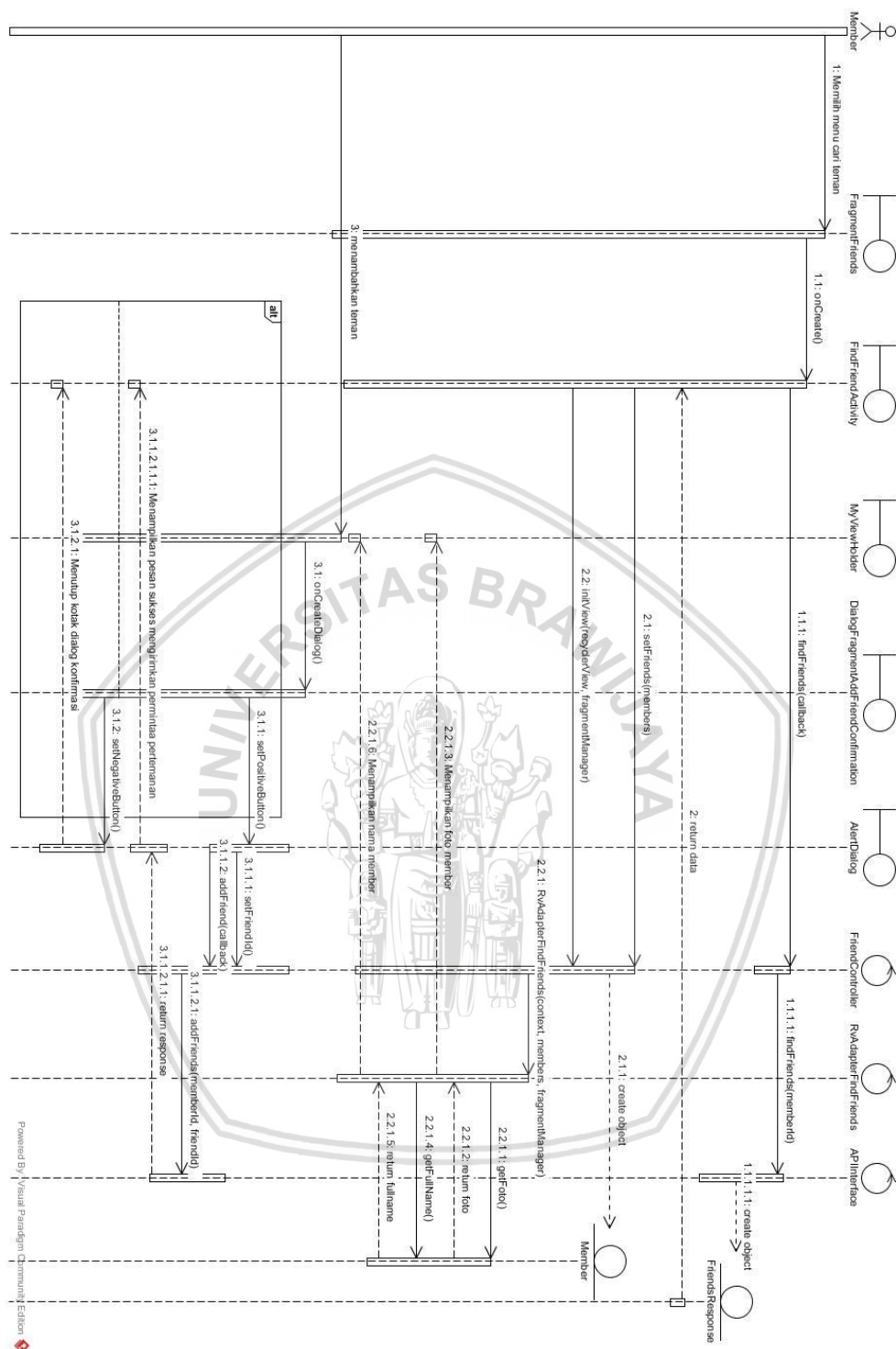
Pada Gambar 5.2 berikut, proses mengajukan tantangan Member menekan tombol challenge pada FragmentFriends, sehingga membuka halaman ChallengeCategoryActivity dan memanggil method `getDataAllCategories()` di kelas `CategoryController`. Selanjutnya `CategoryController` memanggil `getAllCategories` pada kelas `APIInterface` untuk mendapatkan data kategori dari *web service* dan menghasilkan objek `Category`. Selanjutnya data dikembalikan pada kelas `ChallengeCategoryActivity` dan dilakukan inisialisasi tampilan dengan memanggil method `initChallengeCategoryView` pada `CategoryController`. Selanjutnya `CategoryController` memanggil kelas `RvAdapterChallengeCategory` sebagai kelas adapter yang mengambil data foto dan nama kategori dari objek `Category` dan menampilkannya pada kelas `MyViewHolder`. Saat member *memilih* kategori tantangan maka halaman `ChallengeQuestionActivity` muncul dan memanggil method `getDataListQuestions()` di kelas `ChallengeController`. Selanjutnya `ChallengeController` memanggil method `getQuestion()` di kelas `APIInterface` untuk mendapatkan data soal tantangan dari *web service* dan menghasilkan objek `ChallengeQuestionResponse`. Selanjutnya data dikembalikan pada kelas `ChallengeQuestionActivity` dan dilakukan inisialisasi tampilan dengan memanggil method `initChallengeQuestionView()` pada `ChallengeController`. Selanjutnya `ChallengeController` memanggil kelas `RvAdapterChallengeQuestion` sebagai kelas adapter yang mengambil data *title* dari objek `Question` dan menampilkannya pada kelas `MyViewHolder`. Saat Member memilih soal tantangan maka muncul kotak dialog `DialogFragmentChallengeConfirmation` yang akan menampilkan 2 pilihan tombol yaitu *YES* untuk melanjutkan tantangan dan *NO* membatalkan tantangan. Saat Member menekan tombol *YES* maka proses di dalam method `setPositiveButton()` di kelas `AlertDialog` dijalankan, yaitu memanggil method `confirmChallenge()` di kelas `ChallengeController`. Selanjutnya `ChallengeController` memanggil method `getChallengeConfirmation()` dari kelas `APIInterface` untuk mengirim data challenge pada *web service* dan mendapatkan response dalam objek `ChallengeConfirmResponse` yang dikembalikan pada kelas `AlertDialog`. Selanjutnya `AlertDialog` menampilkan halaman `TakeChallengeActivity` yaitu halaman yang digunakan untuk mengerjakan tantangan. Saat Member menekan tombol *NO* maka proses didalam method `setNegativeButton` di kelas `AlertDialog` dijalankan, yaitu menutup kotak dialog konfirmasi.



43

5.1.1.3 Sequence Diagram Mengirim Permintaan Pertemanan

Pada Gambar 5.3 berikut, proses mengirimkan permintaan pertemanan dimulai saat aktor memilih menu cari teman pada FragmentFriends. Selanjutnya aplikasi menjalankan method onCreate() pada halaman FindFriendActivity. Selanjutnya FindFriendActivity memanggil method findFriends() dari kelas FriendController dengan parameter *callback*. Kemudian FriendController memanggil method findFriends() dari kelas APIInterface dengan parameter ID Member. Pemanggilan API menghasilkan *response* berupa object FriendsResponse yang berisi data Member lain. Selanjutnya FindFriendActivity memanggil method setFriends(). Selanjutnya FindFriendActivity memanggil method initView() dari FriendController. Kemudian FriendController melakukan inisiasi kelas RvAdapterFindFriends sebagai kelas adapter. RvAdapterFindFriends mengambil data dari objek Member dengan memanggil method getPhoto() dan getFullName() sehingga objek Member mengembalikan data yang diminta ke kelas MyViewHolder. Kelas MyViewHolder inilah yang akan menampilkan daftar Member lain. Selanjutnya saat aktor menekan tombol tambahkan teman maka kelas RvAdapterFindFriends melakukan inisiasi kelas DialogFragmentAddFriendConfirmation sebagai kelas yang akan menampilkan dialog konfirmasi. Selanjutnya DialogFragmentAddFriendConfirmation menjalankan method onCreateDialog(). Method onCreateDialog() memanggil method setFriendId() untuk melakukan mendapatkan *ID Friend* pada FriendController dan kemudian memanggil method addFriend() dengan parameter *callback*. Selanjutnya FriendController memanggil method addFriends() pada kelas APIInterface dengan parameter *ID Member* dan *ID Friend*. Jika berhasil maka dari pemanggilan API tersebut menghasilkan *response* berarti sukses mengirimkan permintaan pertemanan, sehingga FindFriendActivity menampilkan pesan sukses pada halaman aplikasi.



Gambar 5.3 Sequence Diagram Mengirim Permintaan Pertemanan

5.1.2 Perancangan Class Diagram

Class diagram merupakan sebuah diagram yang menunjukkan seluruh objek yang ada pada sistem yang dinyatakan dalam kelas-kelas serta hubungan antar objek tersebut. Class diagram dari aplikasi pembelajaran pemrograman Java berbasis android ditunjukkan pada Gambar 5.4 berikut.



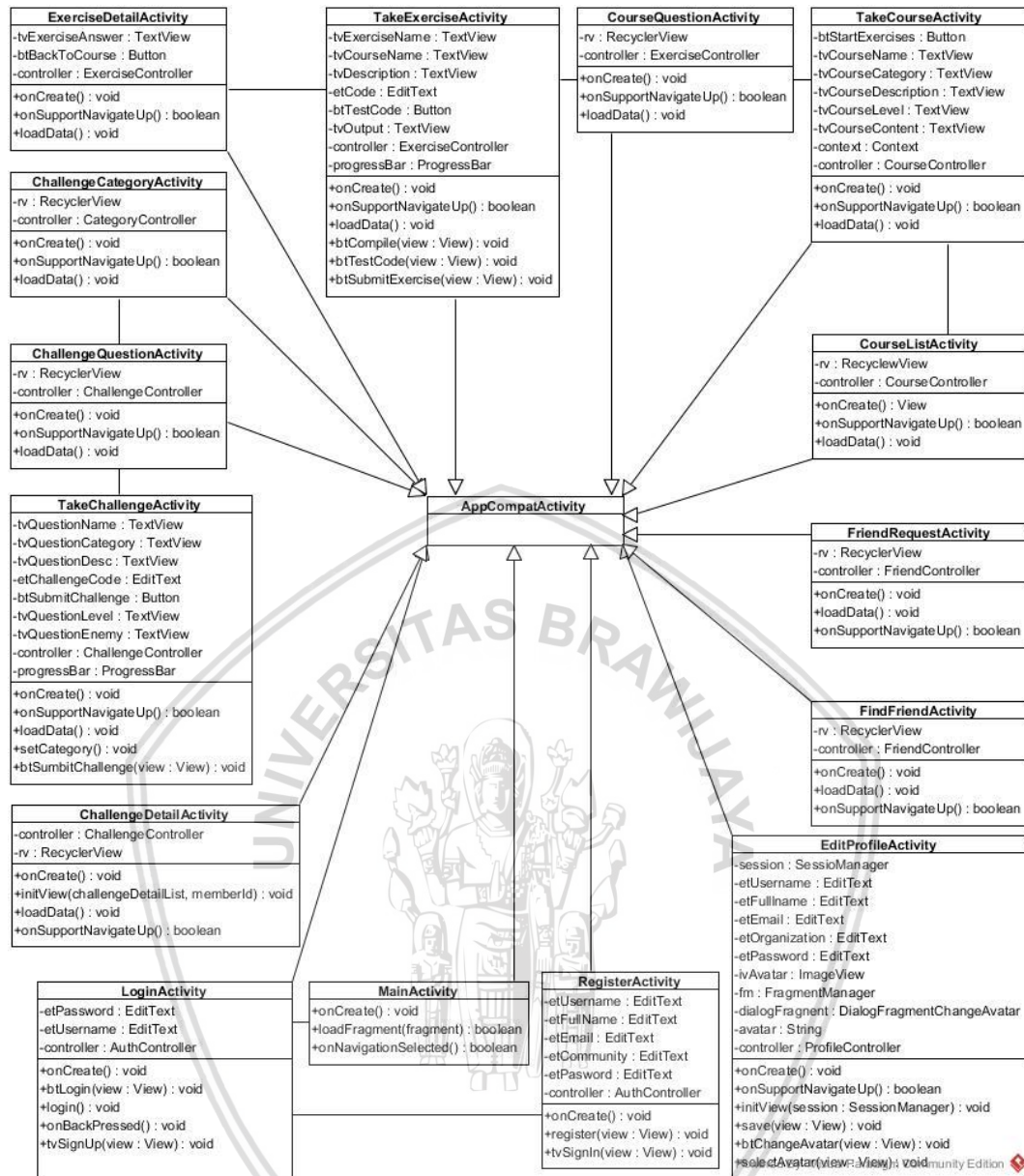
46

Pada Gambar 5.4 diatas, secara umum kelas-kelas dibagi menjadi beberapa jenis kelas berdasarkan *package*, antar lain:

1. *Activity*, merupakan kelas yang berfungsi untuk menampilkan halaman-halaman antarmuka aplikasi kepada pengguna.
2. *Fragment*, merupakan kelas yang memiliki fungsi sama dengan *activity* yaitu menampilkan halaman antarmuka kepada pengguna. Perbedaannya *fragment* menempel pada *activity*, sehingga daur hidup *fragment* bergantung pada daur hidup *activity*.
3. *Controller*, merupakan kelas yang memiliki hubungan dengan *activity* ataupun *fragment* dan *model*. Fungsi *controller* adalah melakukan *request* data pada *server* dan mengembalikan *response* data dalam bentuk objek yang direpresentasikan dalam *model*.
4. *Model*, merupakan kelas yang digunakan untuk merepresentasikan objek-objek yang diperlukan dalam sistem. Kelas model terbagi dalam 2 *package* yaitu *package model* dan *package response*.
5. *Adapter*, merupakan kelas yang digunakan sebagai jembatan baik antara *model* dengan *view* ataupun antara *view* dengan *view*.
6. *API*, merupakan kelas yang digunakan untuk berkomunikasi antara aplikasi android dengan *web service*.

5.1.2.2 Informasi *Class Activity*

Kelas-kelas yang termasuk pada *package activity* dapat dilihat pada Gambar 5.5 berikut.



Gambar 5.5 Informasi *Class Activity*

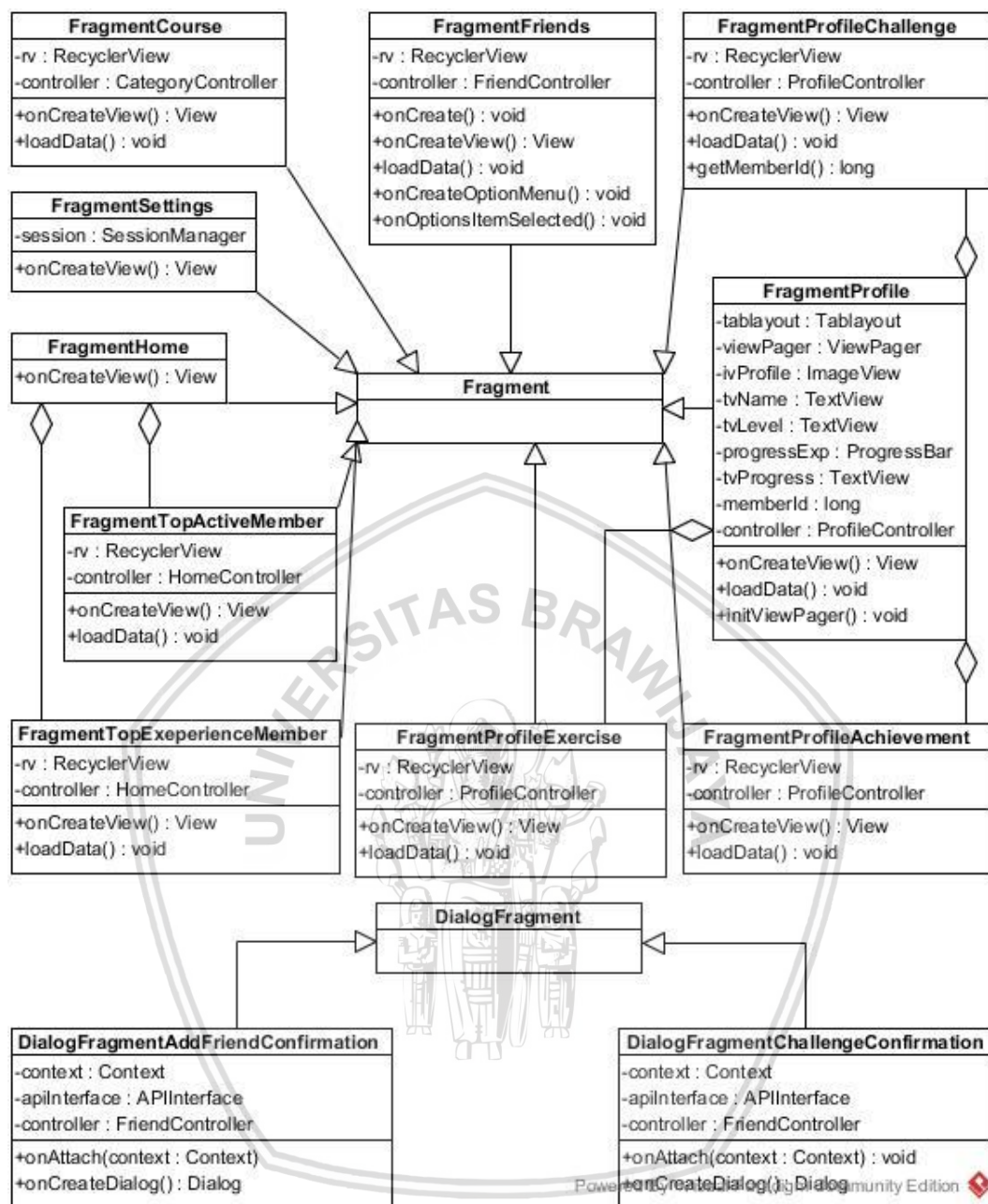
Berdasarkan Gambar 5.5 diatas, terdapat 15 kelas *activity* yang menyusun aplikasi pembelajaran pemrograman Java berbasis android. Keseluruhan *activity* merupakan turunan dari kelas *AppCompatActivity*. Kelas *MainActivity* digunakan menampilkan halaman utama aplikasi, kelas *EditProfileActivity* untuk menampilkan halaman edit profil member, kelas *FindFriendActivity* untuk menampilkan halaman mencari teman baru, kelas *FriendRequestActivity* untuk menampilkan halaman daftar permintaan dari Member lain, kelas *CourseListActivity* untuk menampilkan dan memilih daftar *course* yang tersedia, kelas *TakeCourseActivity* untuk menampilkan halaman materi *course* yang dipilih, kelas *ExerciseQuestionActivity* untuk menampilkan dan memilih daftar soal latihan, kelas *TakeExerciseActivity* untuk menampilkan halaman pengerjaan latihan, kelas *ExerciseDetailActivity* untuk menampilkan halaman detail latihan

yang telah dikerjakan. Kelas `ChallengeCatgeoryActivity` digunakan untuk menampilkan daftar kategori *challenge* yang tersedia, kelas `ChallengeQuestionActivity` digunakan untuk menampilkan dan memilih daftar soal *challenge* berdasarkan kategori yang dipilih, kelas `TakeChallengeActivity` yang digunakan untuk menampilkan halaman pengerjaan tantangan serta kelas `ChallengeDetailActivity` yang digunakan untuk menampilkan rincian *challenge* yang dilakukan ataupun didapatkan oleh Member. Terdapat kelas `LoginActivity` untuk menampilkan halaman *login* dan kelas `RegisterActivity` untuk menampilkan halaman registrasi pengguna.

5.1.2.3 Informasi *Class Fragment*

Kelas-kelas yang termasuk pada *package fragment* dapat dilihat pada Gambar 5.6 berikut.





Gambar 5.6 Informasi Class *Fragment*

Berdasarkan Gambar 5.6 diatas, terdapat 12 kelas yang termasuk dalam *package fragment*. 10 kelas merupakan turunan dari kelas *Fragment* dan 2 kelas merupakan turunan dari kelas *DialogFragment*. Kelas *FragmentHome* yang memiliki 2 kelas turunan yaitu *FragmentTopActiveMember* yang digunakan untuk menampilkan halaman daftar Member dengan peringkat waktu aktif tertinggi dan kelas *FragmentTopExeperienceMember* untuk menampilkan halaman daftar member dengan *level* teratas. Kelas *FragmentProfile* digunakan untuk menampilkan halaman profil member dan memiliki 3 kelas turunan yaitu *FragmentProfileChallenge* untuk menampilkan daftar tantangan yang diajukan dan diperoleh Member, *FragmentProfileAchievement* untuk menampilkan daftar

5.1.2.4 Infomasi *Class Controller*

```

classDiagram
    class ProfileController {
        context: Context
        apiInterface: APIInterface
        -username: String
        -email: String
        -password: String
        -fullName: String
        -community: String
        -avatar: String
        -challenges: List<Challenge>
        -exercises: List<Exercise>
        -achievements: List<Achievement>
        +ProfileController(context: Context, apiInterface: APIInterface)
        +getProfile(callback: Callback<ProfileResponse>): void
        +saveProfileUpdate(callback: Callback<Boolean>): void
        +getFullName(): String
        +setUsername(username: String): void
        +getEmail(): String
        +setEmail(email: String): void
        +getPassword(): String
        +setPassword(password: String): void
        +getFullName(): String
        +setFullName(fullName: String): void
        +getCommunity(): String
        +setCommunity(community: String): void
        +getAvatar(): String
        +setAvatar(avatar: String): void
        +getDataChallenges(callback: Callback<List<Challenge>>): void
        +getExercises(callback: Callback<List<Exercise>>): void
        +getAchievements(callback: Callback<List<Achievement>>): void
        +getChallenges(): List<Challenge>
        +setChallenges(challenges: List<Challenge>): void
        +getExercises(): List<Exercise>
        +setExercises(exercises: List<Exercise>): void
        +getAchievements(): List<Achievement>
        +setAchievements(achievements: List<Achievement>): void
        +getMembers(): long
        +inProfileAchievementView(w: RecyclerView): void
        +inProfileExerciseView(w: RecyclerView): void
        +inProfileAchievementView(w: RecyclerView): void
    }

    class ChallengeController {
        context: Context
        apiInterface: APIInterface
        -intent: Intent
        -bundle: Bundle
        -timer: Stopwatch
        -challengeAnswer: String
        -categories: List<Category>
        -questions: List<Question>
        -challengeDetailWithTypeList: List<ChallengeDetailWithType>
        +ChallengeController(context: Context, apiInterface: APIInterface, intent: Intent)
        +ChallengeController(context: Context, apiInterface: APIInterface, bundle: Bundle)
        +ChallengeController(context: Context, apiInterface: APIInterface)
        +getDataAllCategories(callback: Callback<List<Category>>): void
        +setCategories(categories: List<Category>): void
        +getCategories(): List<Category>
        +getChallengeDetail(callback: Callback<ProfileChallengeDetailResponse>): void
        +inChallengeCategoryView(w: RecyclerView, RecyclerViewChallengeCategory, friendid: long)
        +setChallengeAnswer(challengeAnswer: String): void
        +getMembers(): long
        +inChallengeDetailWithTypeView(w: RecyclerView, RecyclerViewChallengeDetailWithTypeList: List<ChallengeDetailWithType>): void
        +getChallengeDetailWithTypeList(): List<ChallengeDetailWithType>
        +takeChallenge(callback: Callback<TakeChallengeResponse>): void
        +getChallengeDetail(callback: Callback<ChallengeDetailResponse>): void
        +startTimer(): void
        +stopTimer(): void
        +getTimer(): Time
        +getQuestions(): List<Question>
        +setQuestions(questions: List<Question>): void
        +getChallengeAnswer(): String
        +getDetailListQuestions(callback: Callback<SubItemExerciseResponse>): void
        +submitChallenge(callback: Callback<Boolean>): void
        +confirmChallenge(callback: Callback<ChallengeConfirmResponse>): void
        +inChallengeDetailActivityView(w: RecyclerView): void
    }

    class FriendController {
        context: Context
        apiInterface: APIInterface
        -friends: List<Member>
        -friendid: long
        +FriendController(context: Context, apiInterface: APIInterface)
        +getDataAllFriends(callback: Callback<List<Member>>): void
        +inFriends(callback: Callback<FriendsResponse>): void
        +getFriendsRequest(callback: Callback<FriendsResponse>): void
        +getFriendid(): List<Member>
        +setFriends(friends: List<Member>): void
        +getFriendid(): long
        +setFriend(friendid: long): void
        +setFriend(callback: Callback<Boolean>): void
        +inFriendsView(w: RecyclerView)
        +inFindFriendsView(w: RecyclerView, FragmentManager: FragmentManager)
        +inFindRequestsView(w: RecyclerView): void
        +getMembers(): void
    }

    class CourseController {
        context: Context
        apiInterface: APIInterface
        -course: Course
        -intent: Intent
        -courses: List<Course>
        -categories: List<Category>
        -questions: List<Question>
        +CourseController(context: Context, apiInterface: APIInterface, intent: Intent)
        +getDataCourseDetail(callback: Callback<List<CourseReview>>): void
        +getDataCourseDetail(callback: Callback<Course>): void
        +getDataCourseCategories(callback: Callback<List<Category>>): void
        +getDataCourseQuestions(callback: Callback<List<Question>>): void
        +getCourse(): Course
        +setCourse(course: Course): void
        +getCourseList(): List<Course>
        +setCourses(courses: List<Courses>): void
        +getCategories(): List<Category>
        +setCategories(categories: List<Category>): void
        +getQuestions(): List<Question>
        +setQuestions(questions: List<Question>): void
        +inCourseCategoryView(w: RecyclerView): void
        +inCourseListView(w: RecyclerView): void
        +inCourseCourseView(w: RecyclerView): void
    }

    class ExerciseController {
        context: Context
        apiInterface: APIInterface
        -intent: Intent
        -key: String
        -answer: String
        -timer: Stopwatch
        -questionID: int
        -exerciseID: long
        +ExerciseController(context: Context, apiInterface: APIInterface, intent: Intent)
        +getExerciseDetail(callback: Callback<ExerciseDetailResponse>): void
        +takeExercise(callback: Callback<ExerciseResponse>): void
        +submitExercise(callback: Callback<SubmitExerciseResponse>): void
        +testCode(callback: Callback<Boolean>): void
        +compileCode(callback: Callback<Boolean>): void
        +getKey(): String
        +setKey(key: String): void
        +getAnswer(): String
        +setAnswer(answer: String): void
        +getQuestionID(): int
        +setQuestionID(questionID: int): void
        +getExerciseID(): long
        +setExerciseID(exerciseID: long): void
    }

    class AuthController {
        context: Context
        apiInterface: APIInterface
        -username: String
        -password: String
        -email: String
        -fullName: String
        -community: String
        -loginResponse: LoginResponse
        +AuthController(context: Context, apiInterface: APIInterface)
        +login(callback: Callback<LoginResponse>): void
        +createLoginSession(): void
        +getUsername(): String
        +setUsername(username: String): void
        +getPassword(): String
        +setPassword(password: String): void
        +getEmail(): String
        +setEmail(email: String): void
        +getFullName(): String
        +setFullName(fullName: String): void
        +getCommunity(): String
        +setCommunity(community: String): void
        +getLoginResponse(): LoginResponse
        +setLoginResponse(loginResponse: LoginResponse): void
        +register(callback: Callback<Boolean>): void
    }

    class HomeController {
        context: Context
        apiInterface: APIInterface
        -members: List<Member>
        +HomeController(context: Context, apiInterface: APIInterface)
        +getDataTopicActiveMembers(callback: Callback<List<Topic>>): void
        +getDataTopicExperienceMembers(callback: Callback<List<Topic>>): void
        +getMembers(): List<Member>
        +setMembers(members: List<Member>): void
        +inTopicExperienceMemberView(w: RecyclerView): void
        +inTopicActiveMemberView(w: RecyclerView, RecyclerViewExperienceCommunityEditor)
    }

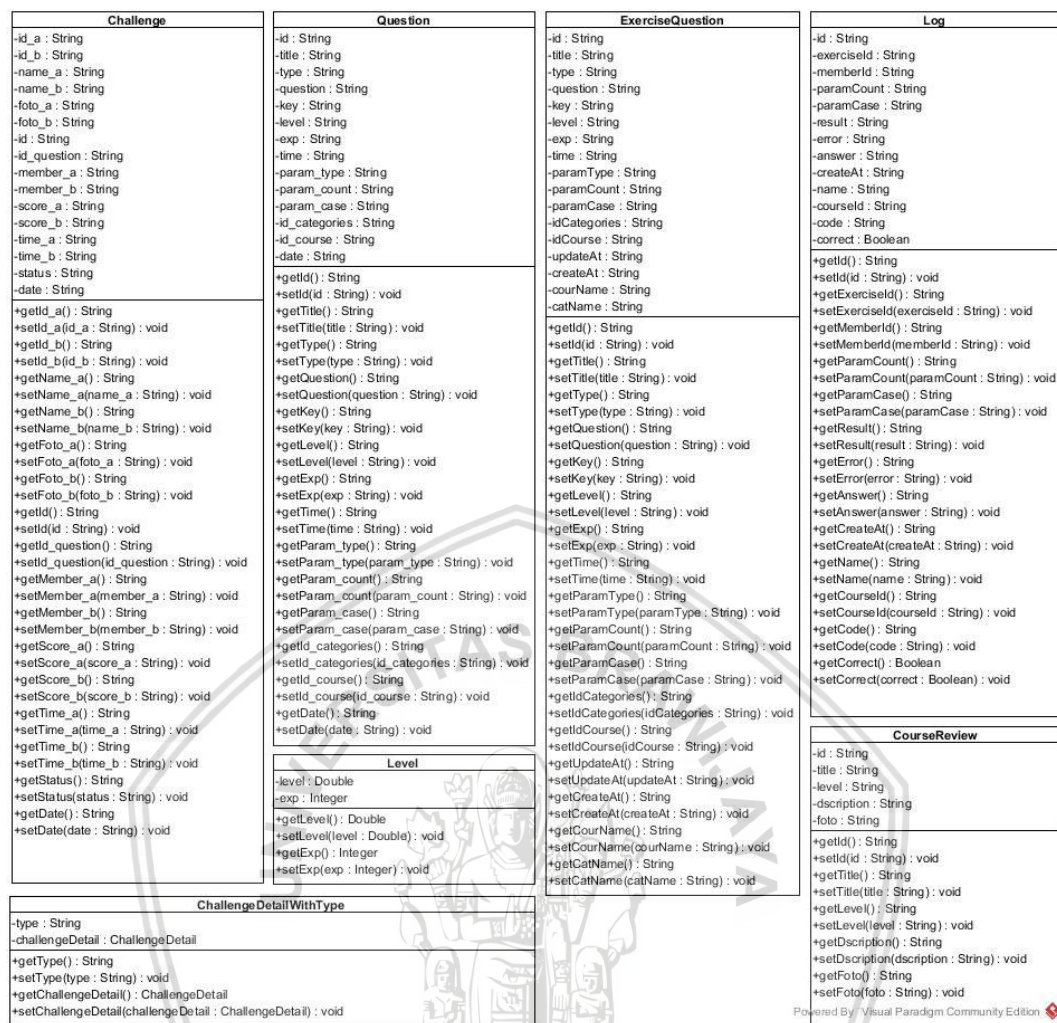
    ProfileController --> ChallengeController
    ProfileController --> FriendController
    ProfileController --> CourseController
    ProfileController --> ExerciseController
    ProfileController --> AuthController
    ProfileController --> HomeController
    
```

Berdasarkan Gambar 5.7 diatas, terdapat 8 kelas yang termasuk dalam kategori controller diantaranya adalah CourseController, ProfileController, HomeController, AuthController, ChallengeController, ExerciseController, CategoryController dan FriendController.

Kelas-kelas yang termasuk pada *package model* dapat dilihat pada Gambar 5.8 dan 5.9, sedangkan *package response* pada Gambar 5.10 berikut.

| ChallengeDetail | Member | Achievement |
|---|--|---|
| -nameA : String -nameB : String -id : String -idQuestion : String -memberA : String -memberB : String -scoreA : String -scoreB : String -timeA : String -timeB : String -status : String -updateAt : String -createAt : String +getNameA() : String +setNameA(nameA : String) : void +getNameB() : String +setNameB(nameB : String) : void +getId() : String +setId(id : String) : void +getIdQuestion() : String +setIdQuestion(idQuestion : String) : void +getMemberA() : String +setMemberA(memberA : String) : void +getMemberB() : String +setMemberB(memberB : String) : void +getScoreA() : String +setScoreA(scoreA : String) : void +getScoreB() : String +setScoreB(scoreB : String) : void +getTimeA() : String +setTimeA(timeA : String) : void +getTimeB() : String +setTimeB(timeB : String) : void +getStatus() : String +setStatus(status : String) : void +getUpdateAt() : String +setUpdateAt(updateAt : String) : void +getCreateAt() : String +setCreateAt(createAt : String) : void | -id : long -username : String -password : String -email : String -fullname : String -exp : String -foto : String -community : String -last_login : String -date : String +getId() : long +setId(id : long) : void +getUsername() : String +setUsername(username : String) : void +getPassword() : String +setPassword(password : String) : void +getEmail() : String +setEmail(email : String) : void +getFullName() : String +setFullName(fullname : String) : void +getExp() : String +setExp(exp : String) : void +getFoto() : String +setFoto(foto : String) : void +getCommunity() : String +setCommunity(community : String) : void +getLast_login() : String +setLast_login(last_login : String) : void +getDate() : String +setDate(date : String) : void | -id : String -id_achievement : String -id_member : String -date : String -name : String -image : String -variable : String -questionResult : String -value : String +getId() : String +setId(id : String) : void +getId_achievement() : String +setId_achievement(id_achievement : String) : void +getId_member() : String +setId_member(id_member : String) : void +getDate() : String +setDate(date : String) : void +getName() : String +setName(name : String) : void +getImage() : String +setImage(image : String) : void +getVariable() : String +setVariable(variable : String) : void +getQuestionResult() : String +setQuestionResult(questionResult : String) : void +getValue() : String +setValue(value : String) : void |
| Category | Exercise | Course |
| -id : String -name : String -foto : String -date : String +getId() : String +setId(id : String) : void +getName() : String +setName(name : String) : void +getFoto() : String +setFoto(foto : String) : void +getDate() : String +setDate(date : String) : void | -id : String -id_member : String -id_question : String -answer : String -score : String -time : String -name : String -date : String +getId() : String +setId(id : String) : void +getId_member() : String +setId_member(id_member : String) : void +getId_question() : String +setId_question(id_question : String) : void +getAnswer() : String +setAnswer(answer : String) : void +getScore() : String +setScore(score : String) : void +getTime() : String +setTime(time : String) : void +getName() : String +setName(name : String) : void +getDate() : String +setDate(date : String) : void | -id : String -name : String -level : String -description : String -content : String -video : String -id_categories : String -cat_name : String -date : String +getId() : String +setId(id : String) : void +getName() : String +setName(name : String) : void +getLevel() : String +setLevel(level : String) : void +getDescription() : String +setDescription(description : String) : void +getContent() : String +setContent(content : String) : void +getVideo() : String +setVideo(video : String) : void +getId_categories() : String +setId_categories(id_categories : String) : void +getCat_name() : String +setCat_name(cat_name : String) : void +getDate() : String +setDate(date : String) : void |

Gambar 5.8 Informasi *Class Model*



Gambar 5.9 Informasi Class Model

Berdasarkan Gambar 5.8 dan 5.9 diatas, terdapat 13 kelas yang termasuk dalam kategori *model*, diantaranya adalah Challenge sebagai representasi dari objek tantangan, Course sebagai representasi dari objek *course*, Log sebagai representasi dari objek *log* latihan, Member sebagai representasi dari objek Member, Question sebagai representasi dari objek soal tantangan, Achievement sebagai representasi dari objek penghargaan, ExerciseQuestion sebagai representasi dari soal latihan, CourseReview sebagai representasi dari objek course yang diambil beberapa atributnya saja, Level sebagai representasi dari objek *level* atau tingkatan member dalam sistem, ChallengeDetail sebagai representasi dari objek detail tantangan, ChallengeDetailWithType sebagai representasi dari objek detail tantangan beserta tipe tantangan, Exercise sebagai representasi dari objek latihan dan Category sebagai representasi dari objek kategori baik kategori *course* ataupun kategori tantangan.

| | | |
|---|--|--|
| SubmitExerciseResponse -answEr : String -idMember : String -idQuestion : String -time : String -score : Integer -idExercise : Integer +getAnswer() : String +setAnswer(answer : String) : void +getIdMember() : String +setIdMember(idMember : String) : void +getIdQuestion() : String +setIdQuestion(idQuestion : String) : void +getTime() : String +setTime(time : String) : void +getScore() : Integer +setScore(score : Integer) : void +getIdExercise() : Integer +setIdExercise(idExercise : Integer) : void | ProfileResponse -newAchievement : List<Object> -loseChallenge : List<Object> -level : Level -member : Member -achievements : List<Achievement> -exercises : List<Exercise> -challenges : List<Challenge> +getNewAchievement() : List<Object> +setNewAchievement(newAchievement : List<Object>) : void +getLoseChallenge() : List<Object> +setLoseChallenge(loseChallenge : List<Object>) : void +getLevel() : Level +setLevel(level : Level) : void +getMember() : Member +setMember(member : Member) : void +getAchievements() : List<Achievement> +setAchievements(achievements : List<Achievement>) : void +getExercises() : List<Exercise> +setExercises(exercises : List<Exercise>) : void +getChallenges() : List<Challenges> +setChallenges(challenges : List<Challenges>) : void | ChallengeQuestionResponse -friendId : String -level : Level -success : Boolean -questions : List<Question> +getFriendId() : String +setFriendId(friendId : String) : void +getLevel() : Level +setLevel(level : Level) : void +getSuccess() : Boolean +setSuccess(success : Boolean) : void +getQuestions() : List<Question> +setQuestions(questions : List<Question>) : void |
| LoginResponse -success : Boolean -info : String -data : List<Member> +getSuccess() : Boolean +setSuccess(success : Boolean) : void +getInfo() : String +setInfo(info : String) : void +getData() : List<Member> +setData(data : List<Member>) : void | ChallengeConfirmResponse -challengeId : Integer +getChallengeId() : Integer +setChallengeId(challengeId : Integer) : void | TakeExerciseResponse -question : ExerciseQuestion -level : Level -index : String +getQuestion() : ExerciseQuestion +setQuestion(question : ExerciseQuestion) : void +getLevel() : Level +setLevel(level : Level) : void +getIndex() : String +setIndex(index : String) : void |
| ChallengeDetailResponse -level : Level -waiting : List<ChallengeDetail> -challenge : List<ChallengeDetail> -history : List<ChallengeDetail> +getLevel() : Level +setLevel(level : Level) : void +getWaiting() : List<ChallengeDetail> +setWaiting(waiting : List<ChallengeDetail>) : void +getChallenge() : List<ChallengeDetail> +setChallenge(challenge : List<ChallengeDetail>) : void +getHistory() : List<ChallengeDetail> +setHistory(history : List<ChallengeDetail>) : void | TakeChallengeResponse -challenge : Question -enemy : Member -idChallenge : String -level : Level +getChallenge() : Question +setChallenge(challenge : Question) : void +getEnemy() : Member +setEnemy(enemy : Member) : void +getIdChallenge() : String +setIdChallenge(idChallenge : String) : void +getLevel() : Level +setLevel(level : Level) : void | RegisterResponse -success : Boolean -info : String +getSuccess() : Boolean +setSuccess(success : Boolean) : void +getInfo() : String +setInfo(info : String) : void |

Gambar 5.10 Informasi *Class Response*

Berdasarkan Gambar 5.10 diatas, terdapat 9 kelas yang ada pada package *Response* yaitu kelas yang digunakan untuk menampung objek yang dihasilkan saat mendapatkan response dari *web service*. Kelas-kelas tersebut diantaranya adalah *SubmitExerciseResponse*, *ProfileResponse*, *ChallengeQuestionResponse*, *LoginResponse*, *ChallengeConfirmResponse*, *TakeExerciseResponse*, *ChallengeDetailResponse*, *TakeChallengeResponse* dan *RegisterResponse*.

5.1.2.6 Informasi *Class Adapter*

Kelas-kelas yang termasuk pada *package adapter* dapat dilihat pada Gambar 5.11 berikut.

| | |
|---|---|
| RvAdapterCourseQuestion - context : Context - questions : List<Question> - courseName : String +RvAdapterCourseQuestion(context : Context, questions : List<Question>, courseName : String) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int | RvAdapterCourseSelection - context : Context - courseReviews : List<CourseReview> - categoryName : String +RvAdapterCourseSelection(context : Context, courseReviews : List<CourseReview>, categoryName : String) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| RvAdapterFindFriends - context : Context - members : List<Member> - fm : FragmentManager - controller : FragmentController +RvAdapterFindFriends(context : Context, members : List<Member>, fm : FragmentManager) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int | RvAdapterChallengeCategory - context : Context - categories : List<Category> - friendId : long +RvAdapterChallengeCategory(context : Context, categories : List<Category>, friendId : long) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| RvAdapterProfileChallengeDetail - context : Context - challenges : List<Challenges> - memberId : long +RvAdapterProfileChallengeDetail(context : Context, challenges : List<Challenges>, memberId : long) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int | RvAdapterProfileChallenge - context : Context - challenges : List<Challenges> - userId : long +RvAdapterProfileChallenge(context : Context, challenges : List<Challenges>, userId : long) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| VPAdapter - fragments : List<Fragment> - fragmentsTitle : List<String> +VPAdapter(fm : FragmentManager) +getItem(position : int) : Fragment +getCount() : int +getPageTitle(position : int) : CharSequence | RvAdapterProfileAchievement - context : Context - achievements : List<Achievement> +RvAdapterProfileAchievement(context : Context, achievements : List<Achievement>) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| RvAdapterFriends - context : Context - members : List<Member> - membersCopy : List<Member> - controller : FriendController +RvAdapterFriends(context : Context, members : List<Member>) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int | RvAdapterProfileExercise - context : Context - exercises : List<Exercises> +RvAdapterProfileExercise(context : Context, exercises : List<Exercises>) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| RvAdapterHome - context : Context - members : List<Member> +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int +RvAdapterHome(context : Context, members : List<Member>) | RvAdapterCourseCategory - context : Context - categories : List<Category> +RvAdapterCourseCategory(context : Context, categories : List<Category>) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |
| RvAdapterChallengeQuestion - context : Context - questions : List<Question> - friendId : long - memberId : long - fm : FragmentManager +RvAdapterChallengeQuestion(context : Context, questions : List<Question>, friendId : long, fm : FragmentManager, memberId : long) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int | RvAdapterFriendRequest - context : Context - members : List<Member> - apiInterface : APIInterface +RvAdapterFriendRequest(context : Context, members : List<Member>) +onCreateViewHolder() : MyViewHolder +onBindViewHolder() : void +getItemCount() : int |

Gambar 5.11 Informasi *Class Adapter*

Dari Gambar 5.11 diatas, terdapat 14 kelas yang termasuk dalam *package adapter*. *Adapter* terbagi menjadi 2 jenis yaitu *RecyclerView Adapter* dan *View Pager Adapter*. *RecyclerView Adapter* adalah sebuah kelas yang digunakan untuk menampilkan mengambil data dari model dan menampilkan data dalam bentuk *list*. Terdapat 13 kelas *RecyclerView Adapter* diantaranya RvAdapterCourseQuestion, RvAdapterProfileChallenge, RvAdapterFindFriends, RvAdapterProfileAchievement, RvAdapterProfileChallengeDetail, RvAdapterChallengeCategory, RvAdapterProfileExercise, RvAdapterHome, RvAdapterCourseCategory, RvAdapterCourseSelection, RvAdapterFriends, RvAdapterFriendRequest, RvAdapterChallengeQuestion. Jenis kedua yaitu *View Pager Adapter* merupakan sebuah kelas yang digunakan untuk mengatur halaman-halaman dalam *tab layout*. Terdapat 1 kelas *View Pager Adapter* yaitu VpAdapter.

5.1.2.7 Informasi *Class API*

Kelas-kelas yang termasuk pada *package API* dapat dilihat pada Gambar 5.12 berikut.

```

<<Interface>>
APIInterface

+register(userName : String, fullName : String, email : String, community : String, password : String) : Call<RegisterResponse>
+login(userName : String, password : String) : Call<LoginResponse>
+getTopActivityMember() : Call<List<Member>>
+getTopExperienceMember() : Call<List<Member>>
+getProfileData(memberId : long) : Call<ProfileResponse>
+getProfileChallenges(memberId : long) : Call<List<Challenge>>
+getProfileExercises(memberId : long) : Call<List<Exercise>>
+getProfileAchievements(memberId : long) : Call<List<Achievement>>
+updateProfile(id : long, username : String, email : String, password : String, fullName : String, community : String, avatar : String) : Call<Boolean>
+getMaleAvatar() : Call<List<String>>
+getFemaleAvatar() : Call<List<String>>
+getAllCategories() : Call<List<Category>>
+getCourseCategories() : Call<List<Category>>
+getCategoryById(categoryId : int) : Call<Category>
+getCourseList(categoryId : int) : Call<List<CourseReview>>
+getCourseDetail(courseId : int) : Call<Course>
+getCourseQuestions(courseId : int) : Call<List<Question>>
+getAllFriends(memberId : long) : Call<AllFriendsResponse>
+findFriends(memberId : long) : Call<FriendsResponse>
+getFriendRequest(memberId : long) : Call<FriendsResponse>
+addFriends(memberId : long, friendId : long) : Call<Boolean>
+acceptFriendRequest(id : long) : Call<Boolean>
+declineFriendRequest(id : long) : Call<Boolean>
+getLevel(memberId : long) : Call<Level>
+getChallengeDetail(memberId : long) : Call<ChallengeDetailResponse>
+getQuestions(memberId : long, friendId : long, categoryId : int) : Call<ChallengeQuestionResponse>
+getChallengeConfirmResponse(memberId : long, friendId : long, categoryId : int, questionId : int) : Call<ChallengeConfirmResponse>
+getTakeChallenge(memberId : long, challengeId : long) : Call<TakeChallengeResponse>
+submitChallenge(memberId : long, answer : String, time : Time, challengeId : long, questionId : int) : Call<Boolean>
+takeExercise(questionId : int, memberId : long) : Call<TakeExerciseResponse>
+compileCode(classname : String, code : String) : Call<String>
+testCode(classname : String, questionId : int, exerciseId : long, answer : String, memberId : long) : Call<String>
+submitExercise(questionId : int, answer : String, time : Time, memberId : long) : Call<SubmitExerciseResponse>
+getExerciseDetail(exerciseId : long, memberId : long) : Call<ExerciseDetailResponse>

```

Powered By Visual Paradigm Community Edition

Gambar 5.12 Informasi Class API

Dari Gambar 5.12 diatas, di dalam package *API* terdapat 1 interface yaitu *APIInterface* yang berisi beberapa method. Method-method tersebut yang nantinya diimplementasikan oleh kelas-kelas yang membutuhkan, yaitu saat sebuah kelas perlu melakukan pemanggilan *API* untuk mendapatkan ataupun mengirimkan data pada *web service*.

5.1.3 Perancangan Komponen

Pada perancangan komponen berisi algoritme masing-masing komponen yang menyusun sistem. Tujuan dari dilakukannya perancangan algoritme adalah untuk memudahkan dalam proses pembuatan kode program yaitu dengan memberikan alur logika fungsionalitas pada masing-masing komponen.

5.1.3.1 Algoritme Melakukan Kompilasi Kode Java

Nama Kelas : TakeExerciseActivity

Nama Operasi : btCompile()

Deskripsi : Algoritme ini digunakan untuk melakukan kompilasi kode program hasil pengerjaan latihan. Operasi ini dipanggil ketika pengguna menekan tombol *compile* pada halaman pengerjaan latihan. Algoritme ini berisi pengambilan data dari API dan menampilkan hasil respon berupa pesan hasil kompilasi kode program pada layar aplikasi bagian *output*. Dalam algortime ini juga melakukan pembangkitan tombol *test code* untuk melakukan percobaan kode Java jika kompilasi berhasil.

Algoritme :

Tabel 5.1 Algoritme Melakukan Kompilasi Kode Java

| | |
|----|-----------------------------------|
| 1 | Set key |
| 2 | Call compile method on Controller |
| 3 | Get response from API |
| 4 | Get body response |
| 5 | If no error message |
| 6 | Print success message to TextView |
| 7 | Enable test code button |
| 8 | Else |
| 9 | Print error message to TextView |
| 10 | Disable test code button |

5.1.3.2 Algoritme Mengajukan Tantangan

Nama Kelas : DialogFragmentChallengeConfirmation

Nama Operasi : onCreateDialog()

Deskripsi : Algoritme ini digunakan sebagai konfirmasi saat mengajukan tantangan. Operasi ini dipanggil ketika pengguna memilih soal di halaman daftar soal tantangan. Algoritme ini berisi pemanggilan API untuk mengirim dan menyimpan data tantangan ke *server* jika Member menekan tombol YES dan aplikasi mengarahkan pada halaman pengerjaan halaman mengerjakan tantangan. Jika member membatalkan tantangan yaitu menekan tombol NO maka kotak dialog konfirmasi ditutup tanpa ada proses apapun.

Algoritme :

Tabel 5.2 Algoritme Mengajukan Tantangan

| | |
|---|--|
| 1 | Initiate AlertDialog Builder |
| 2 | Set dialog title |
| 3 | Set dialog message |
| 4 | Set positive button |
| 5 | Initiate apiInterface |
| 6 | Initiate controller |
| 7 | Call method confirmChallenge() on Controller |
| 8 | Get Reponse from API |

| | |
|----|----------------------------------|
| 9 | Go to TakeChallengeActivity page |
| 10 | Set negative button |
| 11 | Do nothing |
| 12 | Return alertDialog builder |

5.1.3.3 Algoritme Mengirim Permintaan Pertemanan

Nama Kelas : DialogFragmentAddFriendConfirmation

Nama Operasi : onCreateDialog()

Deskripsi : Algoritme ini digunakan untuk mengirim permintaan pertemanan kepada Member lain. Operasi ini dipanggil ketika pengguna menekan tombol yes pada dialog konfirmasi melanjutkan permintaan pertemanan. Algoritme ini berisi pemanggilan API untuk mengirim dan menyimpan data pertemanan pada *server* Jika Member menekan tombol YES. Jika member membatalkan tantangan yaitu menekan tombol NO maka kotak dialog konfirmasi ditutup tanpa ada proses apapun.

Algoritme :

Tabel 5.3 Algoritme Mengirim Permintaan Pertemanan

| | |
|----|---------------------------------------|
| 1 | Initiate AlertDialog Builder |
| 2 | Set dialog title |
| 3 | Set dialog message |
| 4 | Set positive button |
| 5 | Initiate apiInterface |
| 6 | Initiate controller |
| 7 | Call method addFriend() on Controller |
| 8 | Get Reponse from API |
| 9 | Show success message |
| 10 | Finish activity |
| 11 | Set negative button |
| 12 | Do nothing |
| 13 | Return alertDialog builder |

5.1.4 Perancangan Antarmuka

Pada perancangan antarmuka berisi halaman-halaman yang menyusun aplikasi dalam bentuk *low level design*. Berikut adalah hasil rancangan antarmuka aplikasi pembelajaran pemrograman Java berbasis android. Untuk memudahkan dalam menjelaskan setiap bagian dalam rancangan antarmuka diberikan penomoran pada masing-masing halaman antarmuka yang dibahas.

5.1.4.1 Perancangan Antarmuka Halaman Mengerjakan Latihan

Halaman Mengerjakan Latihan digunakan Member untuk menjawab soal latihan. Rancangan antarmuka halaman mengerjakan latihan dijelaskan pada Gambar 5.13 berikut.

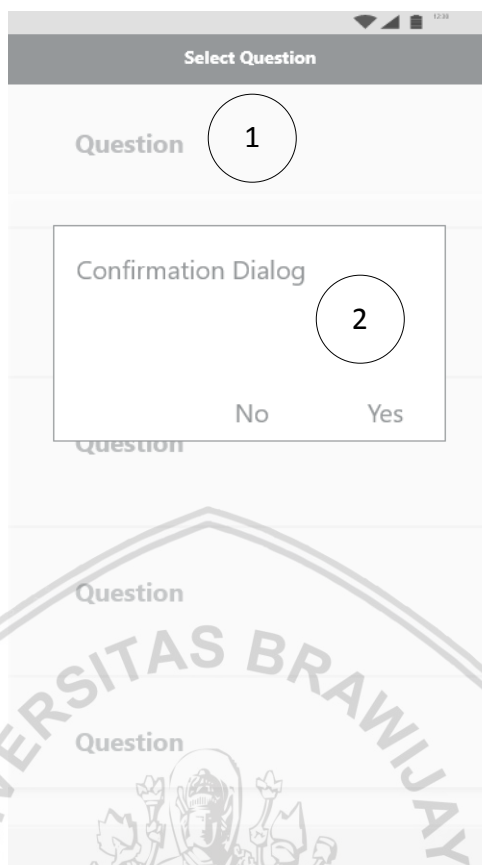
The screenshot shows a mobile application interface titled "Exercise". It contains several components: a header bar with the title, a form for "Exercise Name" (with subtext "Course Name - Level"), a "Question" text area, a "Code Editor" text area, two buttons labeled "Compile" and "Test Code", an "Output" text area, and a "Submit Answer" button at the bottom. Numbered circles (1-6) point to these elements: 1 points to the Exercise Name field, 2 to the Question field, 3 to the Code Editor field, 4 to the Test Code button, 5 to the Output field, and 6 to the Submit Answer button. A large, faint watermark of the Universitas Brawijaya logo is visible in the background.

Gambar 5.13 Rancangan Antarmuka Halaman Mengerjakan Latihan

Gambar 5.13 merupakan rancangan antarmuka halaman mengerjakan latihan. Nomor 1 mendeskripsikan keterangan nama latihan, nama *course* serta *level* soal. Nomor 2 berisi deskripsi soal latihan. Selanjutnya nomor 3 merupakan *field* untuk menuliskan kode program sesuai perintah soal. Nomor 4 merupakan tombol *compile* yang dapat digunakan untuk melakukan kompilasi kode program Java. Nomor 5 merupakan tombol *test code* digunakan untuk melakukan percobaan kode Java untuk mendapatkan keluaran program. Nomor 6 berisi *field output* yang digunakan untuk menampilkan hasil kompilasi ataupun hasil percobaan kode program. Nomor 6 merupakan tombol *submit* yang digunakan untuk mengirimkan data jawaban latihan kepada sistem lain, sehingga sistem lain melakukan operasi tertentu untuk menyimpan data latihan pada database.

5.1.4.2 Perancangan Antarmuka Halaman Mengajukan Tantangan

Halaman Mengajukan Tantangan digunakan Member untuk mengirimkan pengajuan tantangan kepada Member lain. Rancangan antarmuka halaman mengerjakan tantangan dijelaskan pada Gambar 5.14 berikut.

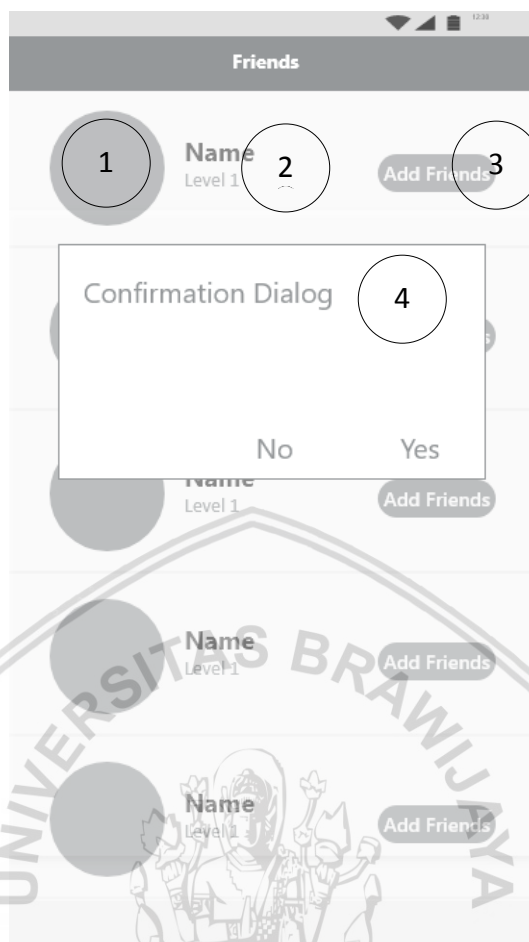


Gambar 5.14 Rancangan Antarmuka Halaman Mengajukan Tantangan

Gambar 5.14 merupakan rancangan antarmuka halaman mengajukan tantangan. Nomor 1 berisi informasi nama soal tantangan. Nomor 2 berisi kotak dialog konfirmasi pengajuan tantangan yang berisi. Pada kotak dialog konfirmasi terdapat 2 tombol yaitu YES dan NO. Tombol YES digunakan untuk melanjutkan tantangan dan menuju ke halaman mengerjakan tantangan, sedangkan tombol NO digunakan untuk membatalkan tantangan.

5.1.4.3 Perancangan Antarmuka Halaman Cari Teman

Halaman cari teman digunakan Member untuk mengirimkan permintaan pertemanan kepada Member lain. Rancangan antarmuka halaman cari teman dijelaskan pada Gambar 5.15 berikut.

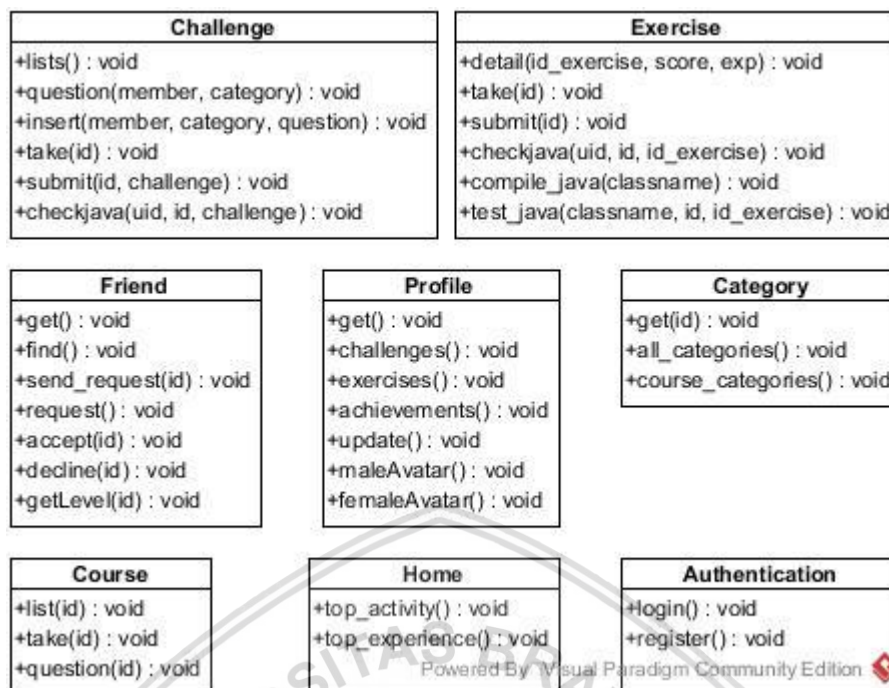


Gambar 5.15 Rancangan Antarmuka Halaman Menemukan Teman Baru

Gambar 5.15 merupakan rancangan antarmuka halaman menemukan teman baru. Nomor 1 berisi foto avatar Member. Nomor 2 berisi keterangan nama Member beserta *level* yang dimiliki. Nomor 3 merupakan tombol untuk mengirimkan permintaan pertemanan. Ketika tombol tersebut ditekan, maka muncul kotak dialog konfirmasi seperti pada nomor 4. Untuk melanjutkan pengiriman permintaan pertemanan dilakukan dengan menekan tombol YES pada kotak dialog tersebut, sedangkan untuk membatalkan pengiriman permintaan pertemanan dengan menekan tombol NO.

5.1.5 Perancangan *Web Service*

Rancangan *web service* yang digunakan sebagai API antara sistem pembelajaran pemrograman Java berbasis website dan aplikasi berbasis android dapat dilihat pada Gambar 5.16 berikut.



Gambar 5.16 Rancangan Web Service

Gambar 5.16 diatas merupakan rancangan web service sistem pembelajaran pemrograman Java. Terdapat 8 kelas yaitu Challenge, Exercise, Friend, Profile, Category, Course, Home dan Authentication. Seluruh kelas berisi method-method yang digunakan untuk menyediakan layanan bagi *client* yaitu aplikasi berbasis android untuk dapat mengakses fungsi-fungsi yang ada pada *server* yaitu sistem berbasis website. Rancangan tersebut nantinya akan diimplementasikan pada sisi *server*.

5.2 Implementasi Sistem

Pada bagian implementasi sistem dijelaskan mengenai proses pembangunan sistem dengan mengacu pada perancangan yang telah dibuat. Tahapan implementasi dimulai dengan mendefinisikan spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam membangun sistem, selanjutnya mendefinisikan implementasi kode program, implementasi basis data serta implementasi antarmuka.

5.2.1 Spesifikasi Sistem

5.2.1.1 Spesifikasi Perangkat Keras

Perangkat keras yang digunakan dalam membangun aplikasi pembelajaran pemrograman Java berbasis android dijelaskan pada Tabel 5.4 berikut.

Tabel 5.4 Spesifikasi Perangkat Keras

| Nama Komponen | Spesifikasi |
|---------------------|--|
| <i>System Model</i> | Lenovo |
| <i>Processor</i> | Intel(R) Core(TM) i3-5005U CPU @ 2.00 GHz 2.00 GHz |
| <i>Memory</i> | 4.00 Gb |
| <i>Display</i> | Intel(R) HD Graphics |
| Solid State Disk | 120 Gb |

5.2.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan dalam membangun aplikasi pembelajaran pemrograman Java berbasis android dijelaskan pada Tabel 5.5 berikut

Tabel 5.5 Spesifikasi Perangkat Lunak

| Nama Komponen | Spesifikasi |
|--------------------------------|---|
| <i>Operating System</i> | Windows 10 Pro 64-bit |
| <i>Programming Language</i> | Java |
| <i>Programming Environment</i> | Android Studio |
| <i>Support Device</i> | Xiaomi Redmi Note 4 - android version 7.1.1 - display 5.5 inchi |

5.2.2 Implementasi Kode Program

Implementasi kode program dilakukan berdasarkan perancangan komponen yang dibuat sebelumnya yang berisi alur logika dan algoritma-algoritma yang menyusun sistem. Algoritma-algoritma dalam bentuk *pseudocode* diubah ke dalam kode progra menggunakan bahasa pemrograman tertentu. Implementasi kode program yang akan ditunjukan sesuai dengan yang ditujukan pada perancangan komponen, yaitu Melakukan Kompilasi, Mengerjakan Tantangan dan Mengirimkan Permintaan Pertemanan.

5.2.2.1 Implementasi Algoritma Melakukan Kompilasi Kode Java

Nama Kelas : TakeExerciseActivity

Nama Operasi : btCompileCode()

Kode Program :

Tabel 5.6 Kode Program Melakukan Kompilasi Kode Java

```

1 controller.setKey(etCode.getText().toString());
2 controller.compileCode(new Callback<String>() {
3     @Override
4     public void onResponse(Call<String> call,
5     Response<String> response) {
6         String result = response.body();
7         if (result.equalsIgnoreCase("")) {
8             String info="Compile Success";
9             tvOutput.setText(info);
10            btTestCode.setEnabled(true);
11        }else {
12            String info = HTMLToString.getString(result);
13            tvOutput.setText(info);
14            btTestCode.setEnabled(false);
15        }
16    }
17
18    @Override
19    public void onFailure(Call<String> call, Throwable t) {
20        String info = "Couldn't access server";
21        Toast.makeText(getApplicationContext(), info,
22        Toast.LENGTH_SHORT).show();
23    }
24 });

```

5.2.2.2 Implementasi Algoritma Mengajukan Tantangan

Nama Kelas : DialogFragmentChallengeConfirmation

Nama operasi : onCreateDialog()

Kode Program :

Tabel 5.7 Kode Program Mengajukan Tantangan

```

1 AlertDialog.Builder builder = new
2 AlertDialog.Builder(getActivity());
3 builder.setTitle("Challenge Confirmation")
4     .setMessage("Continue Challenge ?")
5     .setPositiveButton("Yes", new
6     DialogInterface.OnClickListener() {
7         @Override
8         public void onClick(DialogInterface dialog, int which) {
9             //TODO: get challenge id and go to take
10            apiInterface =
11            APIClient.getClient().create(APIInterface.class);
12            controller = new ChallengeController(context,
13            apiInterface, getArguments());
14            controller.confirmChallenge(new
15            Callback<ChallengeConfirmResponse>() {
16                @Override
17                public void
18                onResponse(Call<ChallengeConfirmResponse> call,
19                Response<ChallengeConfirmResponse> response) {
20                    long challengeId =
21                    response.body().getChallengeId();
22                    Intent intent = new Intent(context,

```

```

23 TakeChallengeActivity.class);
24         intent.putExtra("challengeId", challengeId);
25         context.startActivity(intent);
26     }
27     @Override
28     public void
29 onFailure(Call<ChallengeConfirmResponse> call, Throwable t) {
30         ResponseHandler.showFailureMessage(context);
31     }
32     });
33 }
34 })
35     .setNegativeButton("No", new
36 DialogInterface.OnClickListener() {
37     @Override
38     public void onClick(DialogInterface dialog, int which) {
39         //TODO: nothing
40     }
41 });
42 return builder.create();

```

5.2.2.3 Implementasi Algoritma Mengirimkan Permintaan Pertemanan

Nama Kelas : DialogFragmentAddFriendConfirmation

Nama operasi : onCreateDialog()

Kode Program :

Tabel 5.8 Mengirimkan Permintaan Pertemanan

```

1 AlertDialog.Builder builder = new
2 AlertDialog.Builder(getActivity());
3 builder.setTitle("Add Friends Confirmation")
4     .setMessage("Continue to send friend request ?")
5     .setPositiveButton("Yes", new
6 DialogInterface.OnClickListener() {
7     @Override
8     public void onClick(DialogInterface dialog, int
9 which) {
10         //TODO: Call API to send request
11         apiInterface =
12 APIClient.getClient().create(APIInterface.class);
13         controller = new FriendController(context,
14 apiInterface, getArguments());
15         controller.addFriend(new
16 Callback<Boolean>() {
17     @Override
18     public void onResponse(Call<Boolean>
19 call, Response<Boolean> response) {
20         Toast.makeText(
21             context,
22             "friend request sent",
23             Toast.LENGTH_SHORT
24         ).show();
26         ((FindFriendActivity)
27 context).finish();

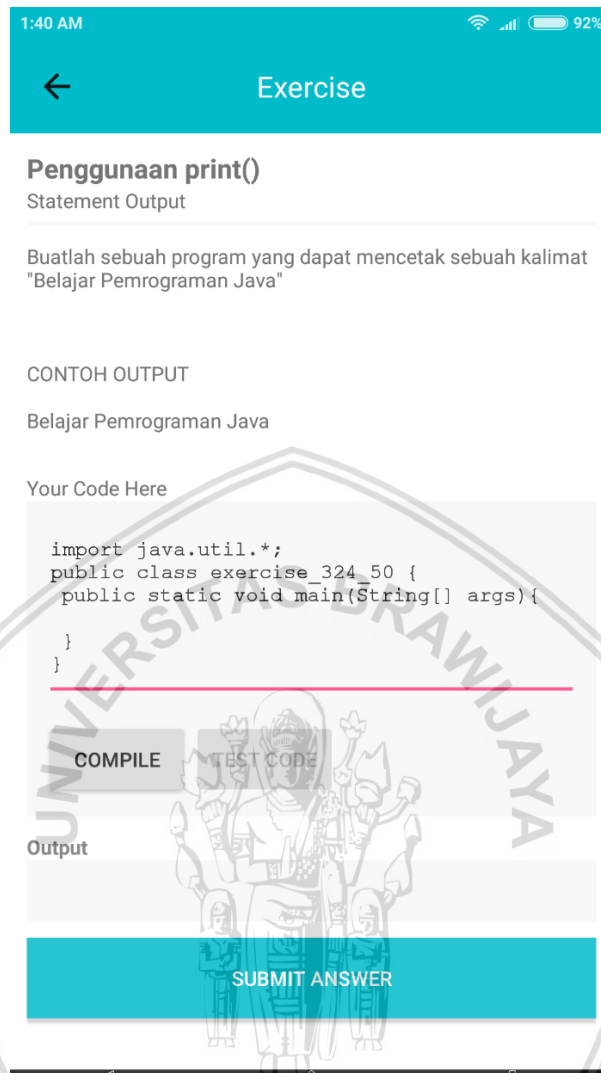
```

```
28         }
29
30         @Override
31         public void onFailure(Call<Boolean>
32 call, Throwable t) {
33
34     ResponseHandler.showFailureMessage(context);
35         }
36         });
37     }
38     })
39     .setNegativeButton("No", new
40 DialogInterface.OnClickListener() {
41         @Override
42         public void onClick(DialogInterface dialog, int
43 which) {
44             //TODO: nothing
45         }
46     });
47     return builder.create();
```

5.2.3 Implementasi Antar Muka

Proses implementasi antarmuka dilakukan dengan mengacu pada perancangan antarmuka yang dibuat sebelumnya. Implementasi antarmuka aplikasi yang ditunjukan sesuai dengan yang ditunjukan pada perancangan antarmuka, yaitu Antarmuka Mengerjakan Latihan, Antarmuka Menjawab Tantangan dan Antarmuka Mengirim Permintaan Pertemanan.

5.2.3.1 Implementasi Antarmuka Halaman Mengerjakan Latihan



Gambar 5.17 Implementasi Antarmuka Halaman Mengerjakan Latihan

Gambar 5.19 diatas menunjukkan halaman antarmuka mengerjakan latihan. Pada bagian *header* halaman terdapat informasi nama latihan dan kategori latihan. Dibawahnya terdapat deskripsi soal berserta contoh outputnya. Selanjutnya terdapat *field* untuk menuliskan kode Java serta tombol *compile* untuk melakukan *kompilasi* dan *test code* untuk melakukan percobaan kode Java. Selanjutnya terdapat *field output* untuk menampilkan hasilnya, baik hasil kompilasi ataupun hasil percobaan kode Java. Dibagian paling bawah terdapat tombol *submit answer* yang digunakan untuk mengirimkan kode Java ke server dan menyimpannya pada *database*.

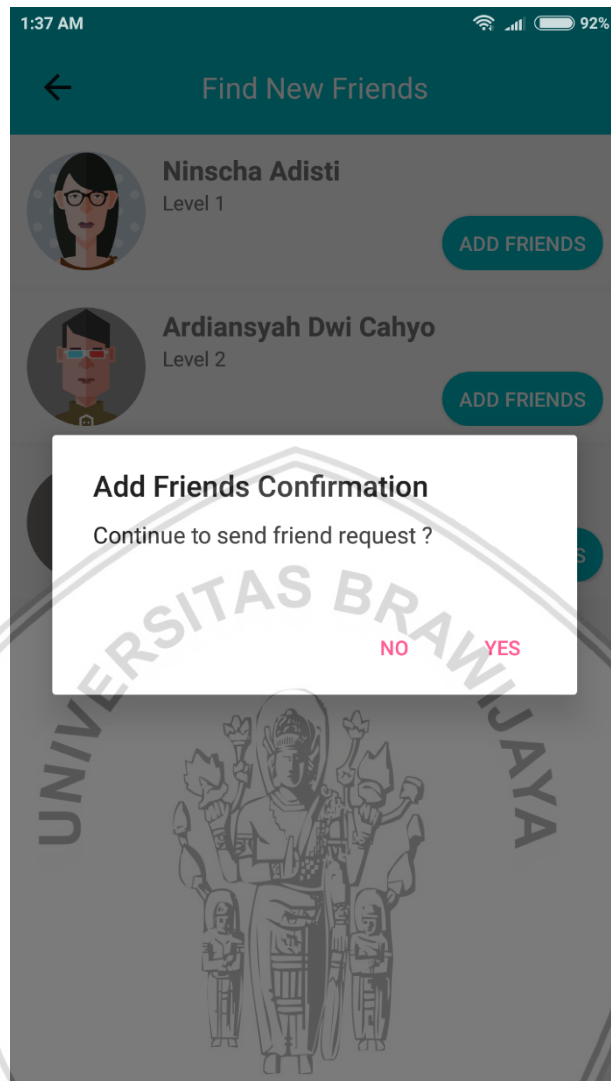
5.2.3.2 Implementasi Antarmuka Halaman Mengajukan Tantangan



Gambar 5.18 Implementasi Antarmuka Halaman Mengajukan Tantangan

Gambar 5.20 diatas menunjukkan halaman antarmuka mengajukan tantangan. Pada halaman ini terdapat daftar soal tantangan yang tersedia berdasarkan kategori tantangan yang dipilih. Saat Member memilih soal, maka kotak dialog konfirmasi akan muncul seperti pada Gambar. Member bisa melanjutkan tantangan dengan menekan tombol YES atau membatalkan tantangan dengan menekan tombol NO.

5.2.3.3 Implementasi Antarmuka Halaman Cari Teman



Gambar 5.19 Implementasi Antarmuka Halaman Cari Teman

Gambar 5.20 diatas menunjukkan halaman antarmuka cari teman yang digunakan untuk menampilkan daftar Member lain yang tidak memiliki hubungan pertemanan. Terdapat informasi foto *avatar*, nama Member dan *level* pada masing-masing data member yang ditampilkan. Terdapat juga tombol *add friend* yang digunakan untuk mengirimkan pertemanan. Saat tombol *add friend* ditekan maka aplikasi menampilkan kotak dialog konfirmasi. Terdapat 2 pilihan tombol yaitu YES yang digunakan untuk melanjutkan dan NO untuk membatalkan pengiriman permintaan pertemanan.

5.2.4 Implementasi *Web Service*

5.2.4.1 Implementasi *method* Melakukan Kompilasi Kode Java

Nama Kelas : Exercise

Nama Operasi : compile_java()

Kode Program :

```

1  $key = $this->input->post('key');
2      $new_data = array(
3          'key' => $key,
4      );
5      $myfile = fopen("./java/test/\".$a.".java", "w") or
6  die("Unable to open file!");
7      fwrite($myfile, $new_data['key']);
8      fclose($myfile);
9      $execute = shell_exec("javac
10 ./java/test/\".$a.".java 2>&1");
11      $execute =
12  str_replace("./java/test/\".$a.".java,"Line ", $execute);
13      $response = $execute;
14      $this->output
15          ->set_status_header(200)
16          ->set_content_type('application/json',
17  'utf-8')
18          ->set_output(json_encode($response,
19  JSON_PRETTY_PRINT));
20          ->_display();
21      exit;

```

5.2.4.2 Implementasi Fungsi Mengajukan Tantangan

Nama Kelas : Challenge

Nama Operasi : insert()

Kode Program :

```

1  $suid = $this->input->post('uid');
2      $data = array(
3          'id_question' => $question,
4          'member_a' => $suid,
5          'member_b' => $member,
6          'status' => 0
7      );
8      $ins_id = $this->m_challenge->add($data);
9      $response['challenge_id'] = $ins_id;
10     $this->output
11         ->set_status_header(200)
12         ->set_content_type('application/json',
13  'utf-8')
14         ->set_output(json_encode($response,
15  JSON_PRETTY_PRINT));
16         ->_display();
17     exit;

```

5.2.4.3 Implementasi Fungsi Mengirimkan Permintaan Pertemanan

Nama Kelas : Friend

Nama Operasi : send_request()

Kode Program :

```
1  $data = array(  
2      'id_member_1' => $this->input->post('uid'),  
3      'id_member_2' => $id  
4  );  
5  $response = $this->m_friend->add($data);  
6  $this->output  
7      ->set_status_header(200)  
8      ->set_content_type('application/json',  
9  'utf-8')  
10     ->set_output(json_encode($response,  
11  JSON_PRETTY_PRINT))  
12     ->_display();  
13  exit;
```



BAB 6 PENGUJIAN

Pengujian dilakukan untuk memastikan seluruh fungsionalitas sistem berjalan sesuai dengan kebutuhan yang telah didefinisikan serta terbebas dari adanya kegagalan. Pengujian yang dilakukan adalah pengujian unit, pengujian integrasi dan pengujian validasi

6.1 Pengujian Unit

Pengujian unit merupakan pengujian yang dilakukan pada komponen terkecil dari sebuah sistem yaitu kelas. Pengujian unit dilakukan untuk memastikan bahwa komponen yang telah diimplementasikan berjalan sesuai dengan fungsionalitasnya. Pengujian unit dilakukan dengan menggunakan metode pengujian *white box* dengan jenis pengujian *basis path*.

6.1.1 Pengujian Unit Kelas TakeExerciseActivity untuk operasi btCompile()

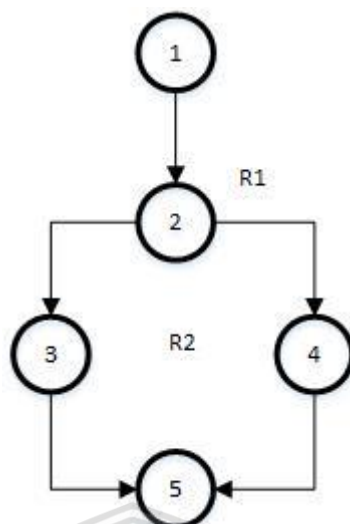
1. Pseudocode

Tabel 6.1 Pseudocode Pengujian Unit Operasi btCompile()

| Pseudocode | No. operasi |
|--|-------------|
| Set Key Call Compile Code on Controller Get Response | 1 |
| If no error message from resposen | 2 |
| Show output compile success Enable test code button | 3 |
| Else (there are error messages from reponse) Show error message Disable test code button | 4 |
| End if | 5 |

2. Basic Path Testing

a. Flow Graph



Gambar 6.1 Flowgraph fungsi *btCompile*

b. *Cyclomatic Complexity*

1. $V(G) = 2 \text{ region (R1, R2)}$
2. $V(G) = 5 \text{ edges} - 5 \text{ nodes} + 2 = 2$
3. $V(G) = 1 \text{ predicate nodes} + 1 = 2$

c. *Independent Path*

1. Jalur 1 = 1-2-3-5
2. Jalur 2 = 1-2-4-5

d. *Prosedur Uji*

| No | Jalur Uji | Prosedur Uji | Expected Result | Result | Status |
|----|-----------|--|---|---|--------|
| 1 | 1 | Menuliskan kode Java dengan benar secara <i>syntax</i> kemudian menekan tombol <i>compile code</i> . | Menampilkan keluaran "Compile Success" | Menampilkan keluaran "Compile Success" | Valid |
| 2 | 2 | Menuliskan kode Java dengan terdapat kesalahan <i>syntax</i> kemudian menekan tombol <i>compile code</i> . | Menampilkan keluaran hasil <i>compile</i> berupa pesan error. | Menampilkan keluaran hasil <i>compile</i> berupa pesan error. | Vaid |

6.1.2 Pengujian Unit Kelas DialogFragmentChallengeConfirmation untuk operasi onCreateDialog()

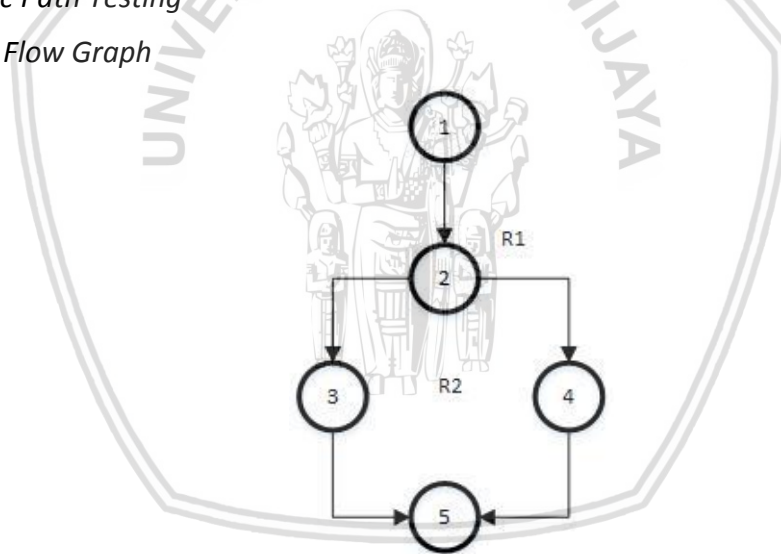
1. Pseudocode

Tabel 6.2 Pseudocode Pengujian Unit Operasi onCreateDialog()

| <i>Pseudocode</i> | No. operasi |
|--|--------------------|
| Initiate AlertDialog Builder Set dialog title Set dialog message Set positive button | 1 |
| Initiate apiInterface Initiate controller Get Reponse from API Go to TakeChallengeActivity page | 2 |
| Set negative button Close dialog | 3 |
| Return alertDialog builder | 4 |
| | 5 |

2. Basic Path Testing

a. Flow Graph



Gambar 6.2 Flowgraph fungsi onCreateDialog

b. Cyclomatic Complexity

1. $V(G) = 2 \text{ region, (R1, R2)}$
2. $V(G) = 5 \text{ edges} - 5 \text{ nodes} + 2 = 2$
3. $V(G) = 1 \text{ predicate nodes} + 1 = 2$

c. Independen Path

1. Jalur 1 = 1-2-3-5
2. Jalur 2 = 1-2-4-5

d. Prosedur Uji

| No | Jalur Uji | Prosedur Uji | Expected Result | Result | Status |
|----|-----------|---|---|---|--------|
| 1 | 1 | Menekan tombol YES pada kotak dialog konfirmasi tantangan | Menuju ke halaman mengerjakan tantangan | Menuju ke halaman mengerjakan tantangan | valid |
| 2 | 2 | Menekan tombol NO pada kotak dialog konfirmasi tantangan | Menutup kotak dialog konfirmasi tanpa melakukan proses apapun | Menutup kotak dialog konfirmasi | valid |

6.1.3 Pengujian Unit Kelas DialogFragmentAddFriendController untuk operasi onCreateDialog()

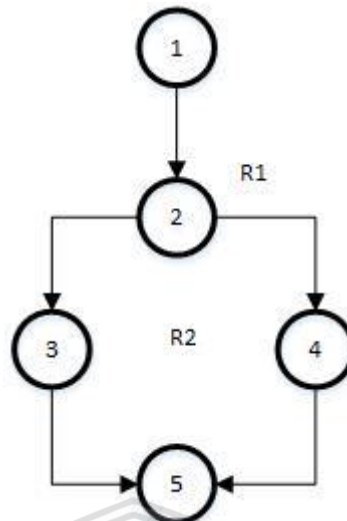
1. Pseudocode

Tabel 6.3 Pseudocode Pengujian Unit Operasi btSubmitChallenge()

| <i>Pseudocode</i> | No. operasi |
|---|-------------|
| Initiate AlertDialog Builder Set dialog title Set dialog message | 1 |
| Set positive button | 2 |
| Initiate apiInterface Initiate Controller Calling addFriend() method on Controller Get Reponse from API Show "friend request sent" message Finish activity | 3 |
| Set negative button Close dialog | 4 |
| Return alertDialog builder | 5 |

2. Basic Path Testing

a. Flow Graph



Gambar 6.3 Flowgraph fungsi *btSubmitChallenge*

b. Cyclomatic Complexity

1. $V(G) = 2 \text{ region (R1, R2)}$
2. $V(G) = 5 \text{ edges} - 5 \text{ nodes} + 2 = 2$
3. $V(G) = 1 \text{ predicate nodes} + 1 = 2$

c. Independent Path

1. Jalur 1 = 1-2-3-5
2. Jalur 2 = 1-2-4-5

d. Prosedur Uji

| No | Jalur Uji | Prosedur Uji | Expected Result | Result | Status |
|----|-----------|---|--|--|--------|
| 1 | 1 | Menekan tombol YES pada kotak dialog konfirmasi menambahkan teman | Menampilkan pesan “permintaan pertemanan terkirim” dan kembali ke halaman <i>friends</i> | Menampilkan pesan “permintaan pertemanan terkirim” dan kembali ke halaman <i>friends</i> | valid |
| 2 | 2 | Menekan tombol NO pada kotak dialog konfirmasi menambahkan teman | Menutup kotak dialog konfirmasi tanpa melakukan proses apapun | Menutup kotak dialog konfirmasi tanpa melakukan proses apapun | valid |

6.2 Pengujian Validasi

Pengujian validasi merupakan pengujian yang dilakukan untuk memastikan bahwa seluruh fungsi pada sistem yang dibangun memiliki kesesuaian kebutuhan yang didefinisikan pada analisis kebutuhan. Pengujian validasi dilakukan dengan menggunakan metode *black box*, yaitu sebuah teknik pengujian yang dilakukan untuk mengetahui keluaran sistem berdasarkan suatu masukan atau aksi tertentu. Pengujian validasi dilakukan berdasarkan skenario *use case* yang dibuat pada tahap analisis kebutuhan.

6.2.1 Pengujian Validasi Fungsi Register

Tabel 6.4 Pengujian validasi fungsi register

| | |
|------------------------------|---|
| Nama Kasus Uji | Register |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Menekan tombol "Sign Up" 3. Mengisi seluruh <i>field</i> yang tersedia dan memasukkan <i>username</i> yang belum terdaftar sebelumnya. 4. Menekan tombol "register" |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan "register berhasil" dan mengarahkan pengguna ke halaman login |
| Hasil | Aplikasi menampilkan pesan "register berhasil" dan mengarahkan pengguna ke halaman login |
| Status | Valid |

a. Kasus Uji Alternatif 1

Tabel 6.5 Pengujian validasi fungsi register alternatif 1

| | |
|------------------------------|---|
| Nama Kasus Uji | Register Alternatif 1 |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Menekan tombol "Sign Up" 3. Mengisi seluruh <i>field</i> yang tersedia dan memasukkan <i>username</i> yang telah terdaftar sebelumnya. 4. Menekan tombol "register" |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan "register gagal, username telah digunakan" |
| Hasil | Aplikasi menampilkan pesan "register gagal, username telah digunakan" |
| Status | Valid |

b. Kasus Uji Alternatif 2

Tabel 6.6 Pengujian validasi fungsi register alternatif 2

| | |
|------------------------------|---|
| Nama Kasus Uji | Register Alternatif 2 |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Menekan tombol "Sign Up" 3. Mengisi tidak mengisi seluruh <i>field</i> yang tersedia 4. Menekan tombol "register" |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan "register gagal, silahkan mengisi seluruh field yang tersedia" |
| Hasil | Aplikasi menampilkan pesan "register gagal, silahkan mengisi seluruh field yang tersedia" |
| Status | Valid |

6.2.2 Pengujian Validasi Fungsi Login**Tabel 6.7 Pengujian validasi fungsi login**

| | |
|------------------------------|--|
| Nama Kasus Uji | Login |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Memasukkan <i>username</i> dan <i>password</i> yang sudah terdaftar sebagai Member. 3. Menekan tombol "login" |
| Hasil yang Diharapkan | Aplikasi mengarahkan pengguna ke halaman <i>home</i> |
| Hasil | Aplikasi mengarahkan pengguna ke halaman <i>home</i> |
| Status | Valid |

a. Kasus Uji Alternatif 1

Tabel 6.8 Pengujian validasi login alternatif 1

| | |
|------------------------------|--|
| Nama Kasus Uji | Login Alterfantif 1 |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Memasukkan <i>username</i> dan <i>password</i> yang belum terdaftar sebagai Member. 3. Menekan tombol "login" |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan "login gagal, username dan password tidak valid" |
| Hasil | Aplikasi menampilkan pesan "login gagal, username dan password tidak valid" |
| Status | Valid |

b. Kasus Uji Alternatif 2

Tabel 6.9 Pengujian validasi login alternatif 2

| | |
|------------------------------|---|
| Nama Kasus Uji | Login Alternatif 2 |
| Prosedur | <ol style="list-style-type: none"> 1. Membuka aplikasi codemaninac 2. Tidak memasukkan <i>username</i> dan <i>password</i> atau salah satunya. 3. Menekan tombol “login” |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “login gagal, masukkan username dan password” |
| Hasil | Aplikasi menampilkan pesan “login gagal, masukkan username dan password” |
| Status | Valid |

6.2.3 Pengujian Validasi Fungsi Melihat Profil

Tabel 6.10 Pengujian validasi fungsi melihat profil

| | |
|------------------------------|---|
| Nama Kasus Uji | Melihat Profil |
| Prosedur | 1. Menekan menu “profil” pada <i>navigation bar</i> |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman berserta informasi profil Member |
| Hasil | Aplikasi menampilkan halaman berserta informasi profil Member |
| Status | Valid |

6.2.4 Pengujian Validasi Fungsi Melakukan Perubahan Profil

Tabel 6.11 Pengujian validasi melakukan perubahan profil

| | |
|------------------------------|--|
| Nama Kasus Uji | Melakukan Perubahan Profil |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “edit profil” pada halaman profil 2. Mengisi field pada form profil yang ingin dirubah 3. Menekan tombol “simpan profil” |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “data profil berhasil disimpan” |
| Hasil | Aplikasi menampilkan pesan “data profil berhasil disimpan” |
| Status | Valid |

6.2.5 Pengujian Validasi Fungsi Melihat Detail Latihan

Tabel 6.12 Pengujian validasi fungsi melihat detail latihan

| | |
|------------------------------|--|
| Nama Kasus Uji | Melihat Detail Latihan |
| Prosedur | <ol style="list-style-type: none"> 1. Mengarahkan <i>tab layout</i> di halaman profil pada tab "exercise" 2. Menekan salah satu daftar latihan |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman berserta informasi detail latihan yang dipilih |
| Hasil | Aplikasi menampilkan halaman berserta informasi detail latihan yang dipilih |
| Status | Valid |

6.2.6 Pengujian Validasi Fungsi Melihat Detail Tantangan

Tabel 6.13 Pengujian validasi fungsi melihat detail tantangan

| | |
|------------------------------|---|
| Nama Kasus Uji | Melihat Detail Tantangan |
| Prosedur | <ol style="list-style-type: none"> 1. Mengarahkan <i>tab layout</i> di halaman profil pada tab "challenge" 2. Menekan tombol detail tantangan |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman berserta informasi detail tantangan |
| Hasil | Aplikasi menampilkan halaman berserta informasi detail tantangan |
| Status | Valid |

6.2.7 Pengujian Validasi Fungsi Melihat Daftar Teman

Tabel 6.14 Pengujian validasi fungsi melihat daftar teman

| | |
|------------------------------|--|
| Nama Kasus Uji | Melihat Daftar Teman |
| Prosedur | 1. Menekan menu "friends" pada <i>navigation bar</i> |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman berserta informasi daftar Member lain yang memiliki hubungan pertemanan |
| Hasil | Aplikasi menampilkan halaman berserta informasi daftar Member lain yang memiliki hubungan pertemanan |
| Status | Valid |

6.2.8 Pengujian Validasi Fungsi Mengirim Permintaan Pertemanan

Tabel 6.15 Pengujian validasi fungsi mengirim permintaan pertemanan

| | |
|-----------------------|--------------------------------|
| Nama Kasus Uji | Mengirim Permintaan Pertemanan |
|-----------------------|--------------------------------|

| | |
|------------------------------|---|
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “find friends” pada halaman “friends” 2. Menekan tombol “add friend” pada salah satu daftar Member 3. Menekan tombol “YES” pada dialog konfirmasi |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “permintaan pertemanan terkirim” |
| Hasil | Aplikasi menampilkan pesan “permintaan pertemanan terkirim” |
| Status | Valid |

a. Kasus Uji Alternatif 1

Tabel 6.16 Pengujian validasi mengirim permintaan pertemanan alternatif 1

| | |
|------------------------------|--|
| Nama Kasus Uji | Mengirim Permintaan Pertemanan Alternatif 1 |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “find friends” pada halaman “friends” 2. Menekan tombol “add friend” pada salah satu daftar Member 3. Menekan tombol “NO” pada dialog konfirmasi |
| Hasil yang Diharapkan | Aplikasi kembali pada halaman cari teman |
| Hasil | Aplikasi kembali pada halaman cari teman |
| Status | Valid |

6.2.9 Pengujian Validasi Fungsi Melihat Daftar Permintaan Pertemanan

Tabel 6.17 Pengujian validasi fungsi melihat permintaan pertemanan

| | |
|------------------------------|---|
| Nama Kasus Uji | Melihat Daftar Permintaan Pertemanan |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “friend request” pada halaman “friends” |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman berserta informasi daftar Member lain yang mengirimkan permintaan pertemanan |
| Hasil | Aplikasi menampilkan halaman berserta informasi daftar Member lain yang mengirimkan permintaan pertemanan |
| Status | Valid |

6.2.10 Pengujian Validasi Fungsi Menerima Permintaan Pertemanan

Tabel 6.18 Pengujian validasi fungsi menerima permintaan pertemanan

| | |
|-----------------------|---|
| Nama Kasus Uji | Merima Permintaan Pertemanan |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “accept friend” pada salah satu |

| | |
|------------------------------|---|
| | Member |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “berhasil menambahkan teman” |
| Hasil | Aplikasi menampilkan pesan “berhasil menambahkan teman” |
| Status | Valid |

6.2.11 Pengujian Validasi Fungsi Mengajukan Tantangan

Tabel 6.19 Pengujian validasi fungsi mengajukan tantangan

| | |
|------------------------------|--|
| Nama Kasus Uji | Mengajukan Tantangan |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “challenge” pada salah satu daftar teman di halaman “friends” 2. Memilih kategori tantangan pada 3. Memilih soal tantangan 4. Menekan tombol “YES” pada dialog konfirmasi |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman mengerjakan tantangan |
| Hasil | Aplikasi menampilkan halaman mengerjakan tantangan |
| Status | Valid |

a. Kasus Uji Alternatif 1

Tabel 6.20 Pengujian validasi fungsi mengajukan tantangan alternatif 1

| | |
|------------------------------|--|
| Nama Kasus Uji | Mengajukan Tantangan Alternatif 1 |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “challenge” pada salah satu daftar teman di halaman “friends” 2. Memilih kategori tantangan 3. Memilih soal tantangan 4. Menekan tombol “NO” pada dialog konfirmasi |
| Hasil yang Diharapkan | Aplikasi menutup dialog konfirmasi dan tetap pada halaman daftar soal |
| Hasil | Aplikasi menutup dialog konfirmasi dan tetap pada halaman daftar soal |
| Status | Valid |

6.2.12 Pengujian Validasi Fungsi Melihat Materi Latihan

Tabel 6.21 Pengujian validasi fungsi melihat materi latihan

| | |
|-----------------------|---|
| Nama Kasus Uji | Melihat Materi Latihan |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan menu “course” pada <i>navigation bar</i> |

| | |
|------------------------------|--|
| | <ol style="list-style-type: none"> Memilih kategori latihan Memilih daftar latihan |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman yang berisi materi latihan |
| Hasil | Aplikasi menampilkan halaman yang berisi materi latihan |
| Status | Valid |

6.2.13 Pengujian Validasi Fungsi Mengerjakan Latihan

Tabel 6.22 Pengujian validasi fungsi mengerjakan latihan

| | |
|------------------------------|---|
| Nama Kasus Uji | Mengerjakan Latihan |
| Prosedur | <ol style="list-style-type: none"> Menekan tombol “mulai latihan” pada halaman materi latihan Memilih soal latihan Menuliskan kode Java Menekan tombol “submit” |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “jawaban berhasil dikumpulkan” dan mengarahkan pengguna ke halaman detail latihan |
| Hasil | Aplikasi menampilkan pesan “jawaban berhasil dikumpulkan” dan mengarahkan pengguna ke halaman detail latihan |
| Status | Valid |

6.2.14 Pengujian Validasi Fungsi Mengerjakan Tantangan

Tabel 6.23 Pengujian validasi fungsi mengerjakan tantangan

| | |
|------------------------------|---|
| Nama Kasus Uji | Mengerjakan Tantangan |
| Prosedur | <ol style="list-style-type: none"> Menuliskan kode Java Menekan tombol “submit” |
| Hasil yang Diharapkan | Aplikasi menampilkan pesan “jawaban berhasil dikumpulkan” dan mengarahkan pengguna ke halaman awal/home |
| Hasil | Aplikasi menampilkan pesan “jawaban berhasil dikumpulkan” dan mengarahkan pengguna ke halaman awal/home |
| Status | Valid |

6.2.15 Pengujian Validasi Fungsi Melakukan Kompilasi Kode Java

Tabel 6.24 Pengujian validasi fungsi melakukan kompilasi kode Java

| | |
|------------------------------|--|
| Nama Kasus Uji | Melakukan Kompilasi Kode Java |
| Prosedur | <ol style="list-style-type: none"> 1. Menuliskan kode Java dengan benar secara <i>syntax</i> 2. Menekan tombol “compile” |
| Hasil yang Diharapkan | Aplikasi menampilkan keluaran “compile success” dan mengaktifkan tombol “test code” |
| Hasil | Aplikasi menampilkan keluaran “compile success” dan mengaktifkan tombol “test code” |
| Status | Valid |

a. Kasus Uji Alternatif 1

Tabel 6.25 Pengujian validasi fungsi melakukan kompilasi kode Java alternatif 1

| | |
|------------------------------|--|
| Nama Kasus Uji | Melakukan Kompilasi Kode Java Alternatif 1 |
| Prosedur | <ol style="list-style-type: none"> 1. Menuliskan kode Java dengan terdapat kesalahan <i>syntax</i> 2. Menekan tombol “compile” |
| Hasil yang Diharapkan | Aplikasi menampilkan keluaran berupa pesan error pada <i>syntax</i> dan menonaktifkan tombol “test code” |
| Hasil | Aplikasi menampilkan keluaran berupa pesan error pada <i>syntax</i> dan menonaktifkan tombol “test code” |
| Status | Valid |

6.2.16 Pengujian Validasi Fungsi Melakukan Percobaan Kode Java

Tabel 6.26 Pengujian validasi fungsi melakukan percobaan kode Java

| | |
|------------------------------|---|
| Nama Kasus Uji | Melakukan Percobaan Kode Java |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “test code” |
| Hasil yang Diharapkan | Aplikasi menampilkan keluaran kode Java |
| Hasil | Aplikasi menampilkan keluaran kode Java |
| Status | Valid |

6.2.17 Pengujian Validasi Fungsi Menjawab Tantangan

Tabel 6.27 Pengujian validasi fungsi menjawab tantangan

| | |
|------------------------------|--|
| Nama Kasus Uji | Menjawab Tantangan |
| Prosedur | <ol style="list-style-type: none"> 1. Menekan tombol “answer challenge” pada halaman detail tantangan |
| Hasil yang Diharapkan | Aplikasi menampilkan halaman mengerjakan tantangan |

| | |
|--------|--|
| Hasil | Aplikasi menampilkan halaman mengerjakan tantangan |
| Status | Valid |

6.2.18 Pengujian Validasi Fungsi Menolak Permintaan Pertemanan

Tabel 6.28 Pengujian validasi fungsi menolak permintaan pertemanan

| | |
|-----------------------|--|
| Nama Kasus Uji | Menolak Permintaan Pertemanan |
| Prosedur | 1. Menekan tombol “decline” pada halaman permintaan pertemanan |
| Hasil yang Diharapkan | Aplikasi menghapus Member dari daftar permintaan pertemanan |
| Hasil | Aplikasi menghapus Member dari daftar permintaan pertemanan |
| Status | Valid |

6.2.19 Pengujian Validasi Fungsi Logout

Tabel 6.29 Pengujian validasi fungsi logout

| | |
|-----------------------|---|
| Nama Kasus Uji | Logout |
| Prosedur | 1. Menekan tombol “logout” pada halaman <i>settings</i> |
| Hasil yang Diharapkan | Aplikasi mengarahkan Member pada halaman <i>login</i> |
| Hasil | Aplikasi mengarahkan Member pada halaman <i>login</i> |
| Status | Valid |

6.2.20 Pengujian Validasi Non Fungsional Compatibility

Tabel 6.30 Pengujian validasi *compatibility* pada sistem operasi android versi 8

| | |
|-----------------------|--|
| Nama Kasus Uji | Pengujian validasi <i>compatibility</i> pada sistem operasi android versi 8 |
| Prosedur | 1. Melakukan instalasi aplikasi <i>codemaniac</i> pada smartphone android versi 8 dengan ukuran layar 6 inchi 2. Membuka aplikasi dan mencoba seluruh fitur yang tersedia |
| Hasil yang Diharapkan | Seluruh fitur pada aplikasi dapat berjalan dengan baik dan antarmuka aplikasi dapat ditampilkan dengan baik |
| Hasil | Seluruh fitur pada aplikasi dapat berjalan dengan baik dan antarmuka aplikasi dapat ditampilkan dengan baik |
| Status | Valid |

Tabel 6.31 Pengujian validasi *compatibility* pada sistem operasi android versi 5

| | |
|------------------------------|--|
| Nama Kasus Uji | Pengujian validasi <i>compatibility</i> pada sistem operasi android versi 5 |
| Prosedur | <ol style="list-style-type: none"> 1. Melakukan instalasi aplikasi <i>codemaniac</i> pada smartphone android versi 8 dengan ukuran layar 4,7 inchi 2. Membuka aplikasi dan mencoba seluruh fitur yang tersedia |
| Hasil yang Diharapkan | Seluruh fitur pada aplikasi dapat berjalan dengan baik dan antarmuka aplikasi dapat ditampilkan dengan baik |
| Hasil | Seluruh fitur pada aplikasi dapat berjalan dengan baik dan antarmuka aplikasi dapat ditampilkan dengan baik |
| Status | Valid |



BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari seluruh proses pengembangan aplikasi mulai dari analisis kebutuhan, perancangan, implementasi dan pengujian yang dilakukan, maka didapatkan kesimpulan sebagai berikut:

1. Proses analisis kebutuhan untuk mengembangkan aplikasi pembelajaran pemrograman Java dilakukan dengan melihat, menganalisa dan menemukan fitur-fitur yang ada pada sistem pembelajaran pemrograman Java berbasis website. Dari proses analisis kebutuhan yang dilakukan, diperoleh 19 kebutuhan fungsional dan 1 kebutuhan non fungsional aplikasi.
2. Perancangan aplikasi pembelajaran pemrograman Java dilakukan dalam 3 tahap, yaitu perancangan arsitektur, perancangan komponen dan perancangan antarmuka. Dari perancangan arsitektur menghasilkan 19 *sequence diagram*, dan 71 *class diagram* diantaranya 31 kelas *boundary*, 22 kelas *controller*, 1 *interface* dan 22 kelas *entity*. Keseluruhan hasil rancangan berhasil diimplementasikan menjadi aplikasi pemrograman pembelajaran pemrograman Java menggunakan *delevopment environtment* android studio.
3. Pengujian aplikasi dilakukan menggunakan 2 strategi pengujian yaitu pengujian unit dan pengujian validasi. Pengujian unit dilakukan menggunakan teknik *basis path* dan berhasil menguji seluruh jalur uji secara valid. Pada pengujian validasi berhasil menguji seluruh kebutuhan fungsional dan non fungsional secara valid.
4. Hasil pengembangan *web service* pada sistem pembelajaran pemrograman Java berbasis website menghasilkan 8 kelas tambahan. Keseluruhan kelas digunakan sebagai media untuk pertukaran data antara *client* yaitu aplikasi android dan *server* yaitu sistem berbasis *website*. *Web service* yang sudah dikembangkan juga dapat digunakan oleh *client* dengan platform lainnya yang ingin melakukan pertukaran data dengan *server*.

7.2 Saran

Dari hasil pengembangan aplikasi pembelajaran pemrograman Java yang atraktif berbasis android, terdapat beberapa saran untuk penelitian selanjutnya diantaranya adalah sebagai berikut:

1. Penambahan fitur *susun kode*, yaitu sudah disediakan potongan-potongan kode program secara acak sebagai untuk menjawab soal yang ada dalam menu tersendiri. Tugas pengguna adalah menyusun potongan-potongan kode program sesuai urutan yang benar. Fitur ini secara tidak langsung dapat membantu pengguna menghafal *syntax* kode program.

DAFTAR PUSTAKA

- Sommerville, I., 2011. *Software Engineering*. 9th ed. United States of America: Pearson Education, Inc.
- Imaduddin, A., Permana, S., 2017. *Menjadi Android Developer Expert*. Bandung: PT. Presentologics.
- Bastari, D.I., Pradana, F. & Priyambadha, B. 2017. *Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website*. Universitas Brawijaya. Malang. Tersedia di <j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/530/217/> [Diakses 15 Januari 2017]
- Kurniawan, T. A. (2018). *Pemodelan Use Case (UML): Evaluasi Terhadap Beberapa Kesalahan Dalam Praktik*. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 77-86.
- Predeira, O., dkk. 2014. *Gamification in Software Engineering – A Systematic Mapping*. Institute of Technology and Information System, University of Castilla-La Mancha, Ciudad Real, Spain.
- Richardson, Amundsend. 2013. *RESTful Web APIs*. United States of America: O'Reilly Media, Inc.
- Mutiawani, V., Subianto, M. & Makruf, M. 2017. *An Android Agricultural Comodity Price Information Application by Utilizing RESTful Web Service*. International Conference on Electrical Engineering and Informatics IEEE.
- <http://tekno.liputan6.com/read/2477796/pengguna-mobile-lebih-suka-pakai-aplikasi-dibanding-browser> [diakes pada 5 Februari 2018]
- <https://developer.android.com/guide/components/fragments.html?hl=id> [diakes pada 23 April 2018]
- <https://developer.android.com/reference/android/app/Activity.html> [diakes pada 23 April 2018]